ISSN 0542-9951

P 1877/94

POLITECHNIKA ŚLĄSKA
Biblioteka Główna
w Gliwicach

2 1994

informatyka

WYDAWNICTWO SIGMA NOT

Cena 35 000 zł

ŚWIATOWY POKAZ TECHNIK INFORMATYCZNYCH

Aktywne i celowe stosowanie najnowszych technik informatycznych wspiera skutecznie realizację podstawowych celów przedsiębiorstwa – dynamikę sprzedaży i stabilną pozycję na rynku. Rozeznanie rozwoju technik informatycznych i możliwości ich zastosowania to konieczność dla współczesnego businessmana. Unikalną i jedyną w skali światowej możliwość w tym względzie daje CeBIT 94, prezentując wszystkie branże technik informatycznych i dziedziny ich praktycznych zastosowań. Tu znajdziesz rozwiązanie wszystkich bez wyjątku problemów organizacyjnych Twojego przedsiębiorstwa. Nie zwlekaj – wizytę na CeBIT 94 (16 – 23 marca 1994) w Hanowerze zaplanuj już dziś!

TECHNIKA INFORMATYCZNA
SIEĆ
CIM (WYTWARZANIE ZINTEGROWANE KOMPUTEROWO)
OPROGRAMOWANIE
TELEKOMUNIKACJA
TECHNIKA BIUROWA
TECHNIKA OPERACJI BANKOWYCH I INNYCH INSTYTUCJI OBROTU PIENIĘŻNEGO
TECHNIKA ZABEZPIECZENIA DANYCH
PRACE BADAWCZE I ROZWOJOWE

HANNOVER 16.–23.03.1994

CeBIT

ŚWIATOWE CENTRUM TECHNIKI BIUROWEJ, INFORMACYJNEJ I TELEKOMUNIKACJI

DEUTSCHE MESSE AG, HANNOVER/GERMANY

Bliższe informacje: MGR ZDZISŁAW A. GEPPERT, PRZEDSTAWICIEL W POLSCE, Deutsche Messe AG, Hannover, ul. Mokotowska 49, 00-542 Warszawa, Tel.: (02) 6600-505, 6600-525, Teleks: 814241 mex pl, Telefaks: (022) 298559

754/93

informatyka

nr 2 1994
Luty
Rok wydania XXX

Miesięcznik
ISSN 0542-9951
INDEKS 36124

BIBLIOTEKA GŁÓWNA Politechniki Śląskiej w Gliwicach
P. 1877/94

W numerze:

Strona

Dedukcyjne bazy danych – *Waldemar Wieczerzycki* 2
Projekt i realizacja obiektowo zorientowanego podsystemu komunikacji z użytkownikiem – *Roman Simiński* 8
Realizacja przedsięwzięć informatycznych z wykorzystaniem metodyki CASE firmy Oracle – *Grażyna Hołodnik-Janczura, Elżbieta Płonka-Szydlak, Ewa Zabawska* 15
Kody kreskowe i automatyczna identyfikacja dla potrzeb EDI – *Anna Kosmacz-Chodorowska* 21
EANCOM – praktyczny przewodnik wdrażania zasad UN/EDIFACT w handlu i przemyśle – *Elżbieta Halas* 25

W najbliższych numerach:

- **Jacek Stochlak charakteryzuje modele i standardy systemów przetwarzania transakcyjnego OLTP.**
- **Wojciech Hałka kreśli perspektywy rozwoju sieci telekomunikacyjnej dla elektronicznej wymiany danych w Polsce.**
- **Maciej Kocięcki prezentuje najważniejsze parametry techniczne sieci pakietowej POLPAK, należącej do Telekomunikacji Polskiej SA oraz przedstawia aspekty ekonomiczne transmisji danych w tej sieci.**
- **Witold D. Lewandowski zajmuje się prywatnymi sieciami telekomunikacyjnymi.**

KOLEGIUM REDAKCYJNE:

mgr Jarosław DEMINET
mgr inż. Piotr FUGLEWICZ
prof. dr hab. Jan GOLIŃSKI
mgr Teresa JABŁOŃSKA (sekretarz redakcji)
Władysław KLEPACZ (redaktor naczelny)
mgr inż. Jan RYŻKO
dr Witold STANISZKIS
dr hab. Kazimierz SUBIETA
dr Zdzisław SZYJEWSKI

PRZEWODNICZĄCY RADY PROGRAMOWEJ:

Prof. dr hab. Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel. 39-14-34

Materiałów nie zamówionych redakcja nie zwraca

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji lub Działu Reklamy i Marketingu
00-950 Warszawa
ul. Mazowiecka 12
telefon: 27-43-66
telefaks: 26-80-16
teleks: 814877

Warunki prenumeraty na 1994 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych.

Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-950 Warszawa, skr. poczt. 1004
konto:
PBK S.A. III 0/Warszawa nr 370015-1573-139-11

Wpłaty na prenumeratę od marca br. przyjmują także wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczyciele na terenie całego kraju

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie ul. Mazowieckiej 12, tel. 26-80-17.

Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 35 000 zł, ulgowa 26 250 zł
Wartość prenumeraty w zł:
Normalna: kwartalna 105 000, półroczna 210 000, roczna 420 000
Ulgowa: kwartalna 78 750, półroczna 157 500, roczna 315 000.

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 512/94

WALDEMAR WIECZERZYCKI
Francusko-Polska Wyższa Szkoła
Nowych Technik Informatyczno-Komunikacyjnych
Poznań

Dedukcyjne bazy danych

W chwili obecnej na rynku komercyjnym ciągle dominują relacyjne bazy danych. Charakteryzują się one prostym modelem danych, stosunkowo wysoką efektywnością i łatwością zastosowań w systemach bankowych, administracyjnych, gospodarki materiałowej i magazynowej, systemach ewidencji ludności, rezerwacji kolejowej i hotelowej itp. Próby zastosowań relacyjnych baz danych do szeroko rozumianego wspomagania podejmowania decyzji wykazały potrzebę rozszerzenia modelu relacyjnego o nowe elementy, ułatwiające modelowanie bardziej skomplikowanej w tych zastosowaniach rzeczywistości, lub wymiany modelu relacyjnego na inny model o podwyższonym stopniu ekspresji. Pierwszy kierunek zaowocował pojawieniem się tak zwanych systemów postrelacyjnych, natomiast drugi – systemów obiektowych i dedukcyjnych.

Obiektowe bazy danych powstały w wyniku połączenia paradygmatu obiektowego, powszechnie stosowanego w nowoczesnych językach programowania, takich jak Smalltalk, czy C_{++}, z mechanizmami klasycznych baz danych. Obiektowe bazy danych dobrze nadają się do wspomagania projektowania (CAD), wspomagania zarządzania (CAM) i wspomagania inżynierii oprogramowania (CASE). Dotychczas zaprojektowano kilkanaście obiektowych systemów baz danych. Niektóre z nich stały się już produktami komercyjnymi. Najbardziej znane obiektowe bazy danych to GemStone, Orion, Ode, O2, Ontos, ObjectStore.

Analogicznie, dedukcyjne bazy danych powstały w wyniku połączenia mechanizmów wnioskowania, stosowanych w językach programowania deklaratywnego, których głównym reprezentantem jest Prolog, z mechanizmami klasycznych baz danych. Dedukcyjne bazy danych dobrze nadają się do systemów eksperckich i systemów wspomagania decyzji. Najbardziej znane dedukcyjne bazy danych to EKS [1], LDL [7], Aditi [8], Coral [5] i Glue-NAIL [4].

Oba, początkowo niezależne, kierunki rozwoju baz danych są aktualnie łączone. Wynikiem tego procesu są eksperymentalne systemy nazywane **dedukcyjnymi obiektowymi bazami danych** [6].

Obiektowe bazy danych zostały przedstawione w [9]. Celem tego artykułu jest przybliżenie dedukcyjnych baz danych. W kolejnych rozdziałach omówiono model danych systemów dedukcyjnych, ich podstawowe funkcje, więzy integralności oraz realizację zapytań. W podsumowaniu przedstawiono różnice między dedukcyjnymi bazami danych i systemami programowania w logice – z jednej strony, oraz między dedukcyjnymi i relacyjnymi bazami danych – z drugiej strony.

Model danych

U podstaw każdego systemu bazy danych leży **model danych** [3], który jest zbiorem pewnych abstrakcyjnych pojęć umożliwiających opis fragmentu rzeczywistości modelowanej w bazie danych, interpretację oraz zmianę jej aktualnego stanu. W modelu danych wyróżnia się:

- **struktury danych**, odpowiadające statycznym własnościom świata rzeczywistego, takie jak typy modelowanych encji (ang. *entity*) oraz powiązania między encjami;
- **operacje**, modelujące dynamiczne własności encji;
- **więzy integralności** (ang. *integrity constraints*), wykluczające niedozwolone stany encji oraz niepoprawne związki między encjami.

W dedukcyjnych bazach danych do opisu danych używa się języka logiki pierwszego rzędu, natomiast operowanie na danych sprowadza się do wartościowania formuł logicznych [2]. Takie podejście prowadzi do następujących korzyści:

- języki logiki powszechnie charakteryzują się dobrze zrozumiałą semantyką;
- języki logiki są narzędziami umożliwiającymi jednorodne opisywanie faktów i reguł rządzących modelowaną rzeczywistością, więzów integralności oraz kierowanych do bazy danych zapytań;
- dojrzała teoria leżąca u podstaw języków logiki ułatwia rozwiązywanie wielu problemów związanych z efektywnym zarządzaniem bazą danych, np. obsługą **wartości pustych** (ang. *nil values*) lub **nieokreślonych danych** (ang. *indefinite data*);
- pojedyncze reguły logiczne, ze względu na wysoki stopień ekspresji, mogą zastąpić całe zbiory jawnie definiowanych faktów; z jednej strony – upraszcza to proces modelowania encji, z drugiej strony – zwiększa czytelność i ułatwia zrozumienie wybranego sposobu opisu rzeczywistości;
- model danych dla dedukcyjnych baz danych może być traktowany jako uogólnienie modelu danych dla relacyjnych baz danych.

W celu formalnego zdefiniowania dedukcyjnej bazy danych wprowadzimy najpierw pojęcia klauzuli i zapytania bazy danych.

Klauzula bazy danych (ang. *database clause*) jest formułą logiczną następującej postaci:

$$A_1 \vee \ldots \vee A_m \leftarrow L_1 \wedge \ldots \wedge L_n$$

gdzie $m \geqslant 1$, $n \geqslant 0$, A_i jest atomem oraz L_j jest literałem. Część klauzuli występująca po jej lewej stronie jest nazywana **głową** klauzuli (ang. *head*), natomiast część występująca po prawej stronie **ciałem** klauzuli (ang. *body*). **Reguła** (ang. *rule*) jest klauzulą z niepustym ciałem (tj. $n > 0$), natomiast **fakt** (ang. *fact*) jest klauzulą pozbawioną ciała.

Zapytanie bazy danych, lub krótko: **zapytanie** (ang. *query*) jest formułą logiczną postaci:

$L_1 \wedge ... \wedge L_n$

gdzie $n \geqslant 1$ oraz L_i jest literałem.

Dedukcyjna baza danych w postaci ogólnej jest definiowana jako para D, L, w której D jest skończonym zbiorem klauzul, a L jest językiem logiki pierwszego rzędu. Zakłada się ponadto, że L zawiera co najmniej dwa typy symboli umożliwiających reprezentowanie stałych (ang. *constant*) i predykatów (ang. *predicate*).

W świetle powyższej definicji, relacyjna baza danych jest traktowana jako szczególny przypadek dedukcyjnej bazy danych D, L, w której D zawiera wyłącznie fakty.

Fakty dedukcyjnych baz danych są wystąpieniami tak zwanych **predykatów bazowych** (ang. *base predicates*). Odpowiadają one w pewnym stopniu krotkom relacyjnych baz danych, będących wystąpieniami relacji. Podstawowa różnica polega na udostępnieniu w dedukcyjnych bazach danych znacznie szerszego zestawu typów prostych (dziedzin) dla atrybutów faktów. W części dedukcyjnych baz danych udostępniono ponadto konstruktory zbiorowe i listowe, umożliwiające definiowanie atrybutów zagregowanych.

Reguły, będące wystąpieniami tak zwanych **predykatów wirtualnych** (ang. *virtual predicates*), umożliwiają wywodzenie nowych faktów (wirtualnych) na podstawie faktów przechowywanych w bazie danych (wystąpień predykatów bazowych) oraz innych faktów wirtualnych, otrzymanych w wyniku wcześniejszego zastosowania reguł innych predykatów. Reguły umożliwiają ponadto specyfikę więzów integralności, ograniczających zbiór możliwych stanów bazy danych.

W niektórych szczególnych sytuacjach predykaty wirtualne mogą być **materializowane** (ang. *materialized*), to znaczy, że wywiedzione z nich fakty mogą zostać w sposób jawny zapamiętane w bazie danych tak, jakby były wystąpieniami odpowiedniego predykatu bazowego. Materializacja predykatów umożliwia zwiększenie efektywności systemu. Dotyczy to, po pierwsze – predykatów, do których często odwołują się zapytania (bezpośrednio lub pośrednio) kierowane przez użytkowników (aplikacje) bazy danych. Po drugie, materializowane mogą być również predykaty, które wyznaczają bardzo liczne zbiory wirtualnych faktów. W obu sytuacjach nie ma zwykle sensu wielokrotne stosowanie tych samych reguł w celu wywiedzenia tej samej informacji. Zmaterializowane predykaty wymagają jednak nowych mechanizmów systemowych, polegających na ich dynamicznym uaktualnianiu wraz ze zmianami faktów predykatów bazowych, do których się odwołują. Podobna aktualizacja jest konieczna w przypadku modyfikacji reguł predykatów wirtualnych, które były użyte w pierwotnym procesie wywodzenia faktów zmaterializowanego predykatu.

Poza predykatami bazowymi i wirtualnymi, większość dedukcyjnych baz danych oferuje tak zwane **predykaty zewnętrzne** (ang. *external predicates*), zdefiniowane poza systemem, często za pomocą klasycznych imperatywnych języków programowania. Predykaty zewnętrzne umożliwiają sprawdzenie, czy między argumentami zachodzą określone związki (np. równości, większości), realizację operacji arytmetycznych na argumentach predykatów, konwersję typów argumentów, wymianę informacji z operatorem itp. Predykaty zewnętrzne nie mogą odwoływać się do funkcji systemu dedukcyjnej bazy danych, w szczególności nie mogą odczytywać i uaktualniać danych.

Języki dedukcyjnych baz danych są zwykle mutacjami języka Prolog. Składają się z dwóch rozłącznych komponentów: deklaratywnego i proceduralnego. **Komponent deklaratywny** jest odpowiednikiem języka definiowania danych relacyjnych baz danych (ang. *Data Definition Language* – DDL). Umożliwia on definiowanie reguł i ograniczeń integralnościowych przechowywanych w bazie danych. Jego deklaratywność polega na tym, że użytkownik nie jest świadomy sposobu, w jaki system wartościuje reguły i ograniczenia podczas współbieżnego wykonywania kolejnych transakcji. Użytkownik również nie ma możliwości wymuszenia określonej procedury wartościowania reguł i ograniczeń integralnościowych, np. kolejności wartościowania elementów reguł złożonych. **Komponent proceduralny** jest odpowiednikiem języka manipulowania danymi relacyjnych baz danych (ang. *Data Manipulation Language* – DML). Umożliwia przygotowanie aplikacji bazy danych, przechowywanych poza bazą danych, oraz wcześniej wspomnianych predykatów zewnętrznych.

Podstawowe funkcje systemu zarządzania dedukcyjną bazą danych

Aplikacje operujące na faktach i regułach bazy danych są zorganizowane w postaci zbiorów elementarnych jednostek interakcji z systemem zarządzania dedukcyjną bazą danych nazywanych **transakcjami**. Transakcja jest sekwencją logicznie powiązanych operacji. Zgodnie z klasycznymi bazami danych ma ona następujące własności:

- **atomowość** – oznaczającą, że operacje transakcji muszą być wykonywane w całości lub w całości wycofane,
- **trwałość** – gwarantującą, że w przypadku pomyślnego zakończenia transakcji wszystkie wprowadzone przez nią zmiany do bazy danych nie zostaną później utracone, nawet w przypadku awarii systemu, oraz
- **spójność** – polegającą na tym, że transakcja odwzorowuje spójny stan bazy danych, to jest taki, w którym spełnione są wszystkie zdefiniowane w niej więzy integralności, w inny spójny stan; w trakcie realizacji transakcji spójność dedukcyjnej bazy danych może być przejściowo naruszona.

System zarządzania dedukcyjną bazą danych powinien charakteryzować się własnościami, które zapewnią bezpieczny i efektywny dostęp do bazy danych wielu użytkownikom. W związku z tym system musi realizować następujące funkcje:

- ochrony spójności i trwałości danych,
- zarządzania współbieżnością transakcji,
- autoryzacji dostępu do danych,
- fizycznego zarządzania danymi,
- weryfikacji więzów integralności,
- optymalizacji wykonywania operacji, w szczególności procesu wnioskowania.

Cztery pierwsze spośród wyżej wymienionych funkcji są implementowane przez adaptację i ewentualne rozszerzenia rozwiązań powszechnie stosowanych w relacyjnych bazach danych. Weryfikacja więzów integralności oraz optymalizacja wykonywania operacji wymagają nowych rozwiązań, co wynika ze znacznego wzbogacenia mechanizmów definiowania więzów integralności oraz zapytań kierowanych do bazy danych, w stosunku do możliwości oferowanych przez relacyjne bazy danych. Przypomnijmy bowiem, że więzy integralności są dowolnymi klauzulami logiki pierwszego rzędu, natomiast zapytania, poza bezpośrednim odwoływaniem się do predykatów bazowych, mogą również odwoływać się do predykatów wirtualnych, opisanych za pomocą reguł wnioskowania.

Klasyczne funkcje systemów zarządzania bazami danych są powszechnie znane i nie wymagają dalszego komentarza.

W związku z tym przedstawimy teraz specyfikę dedukcyjnych baz danych oraz omówimy niektóre charakterystyczne dla nich funkcje.

Przykładowa baza danych

W celu zilustrowania wybranych funkcji dedukcyjnych baz danych rozważmy przykładową magazynową bazę danych, w której zdefiniowano cztery predykaty bazowe:

```
towary_podst (Towar, Dostawca, Koszt)
kompozycja (Towar, Towar_składowy, Liczba_wyst)
stan (Towar, Liczba_egz)
zaufani_dostawcy (Dostawca).
```

Predykat *towary_podst* opisuje przechowywane w magazynie towary, ich dostawców i koszty jednostkowe; predykat *kompozycja* łączy towary złożone z ich komponentami, które mogą występować wielokrotnie; predykat *stan* określa aktualnie dostępną w magazynie liczbę egzemplarzy danego towaru; predykat *zaufani–dostawcy* opisuje tych dostawców, którzy, z określonych powodów, zostali uznani jako „zaufani".

Do opisu predykatów i reguł stosujemy notację języka Prolog: nazwy stałych rozpoczynamy małymi literami, natomiast nazwy zmiennych – dużymi. Łączniki logiczne **and** i **or** uzupełniamy o dwa nowe: **farall** i **exists**, będące odpowiednikami występujących w logice pierwszego rzędu kwantyfikatorów: ogólnego i szczegółowego. Poniższa reguła wyznacza towary podstawowe, których koszt jest niższy od określonego kosztu progowego:

```
tani_towar (Towar, Dostawca, Koszt, Próg) ←
        towary_podst (Towar, Dostawca, Koszt) and
        Próg > Koszt.

Natomiast zapytanie
        find tani_towar (opona, Dostawca, Koszt, 200)
```

odszukuje dostawców opon oferujących cenę poniżej dwustu tysięcy zł.

Ciało reguły definiującej wirtualny predykat *tani_towar* odwołuje się wyłącznie do predykatu bazowego *towary_podst* oraz do zmiennej *Koszt*, która musi być obsadzona (ang. *instantiated*) w zapytaniu. W ogólności, w ciałach reguł można odwoływać się do wielu predykatów, zarówno bazowych, jak i wirtualnych, stosując dodatkowo kwantyfikatory ogólne i/lub szczegółowe. W poniższym przykładzie wyznaczamy towary złożone, które mogą być wyprodukowane z dostępnych w magazynie towarów podstawowych:

```
wytwarzalne_towary (Towar) ←
        forall [Komponent, Liczba_wyst]:
                (towar_komponent_liczba(Towar, Komponent, Liczba_wyst) and
                towary_podst(Komponent, _, _)
                )
        → dostępny_towar(Komponent, Liczba_wyst)

dostępny_towar(Towar, Liczba) ←
        stan (Towar, Liczba_wyst) and
        Liczba_wyst >= Liczba.
```

Predykat wirtualny *towar_komponent_liczba* określa liczbę wystąpień komponentu *Komponent* w towarze złożonym *Towar*, uwzględniając jego całą hierarchię kompozycji.

Operacje na grupach faktów

Dedukcyjne bazy danych umożliwiają podział faktów, pobranych z predykatów bazowych lub wywiedzionych z predykatów wirtualnych, na grupy według wartości określonych atrybutów. Następnie, na grupach tych można operować za pomocą tzw. **funkcji zagregowanych** (ang. *aggregate functions*), predefiniowanych w systemie.

W poniższym przykładzie zdefiniowano predykat wirtualny *min_koszt*, który wyznacza najniższy koszt określonego towaru, potencjalnie pochodzącego od wielu dostawców. Odwołanie do predykatu *towary_podst* powoduje wyszukanie zbioru faktów dotyczących towaru, którego nazwa jest argumentem odwołania. Fakty te są następnie grupowane zgodnie z listą zmiennych grupujących, a następnie zostaje wyznaczona wartość funkcji zagregowanej (w przykładzie wartość funkcji *min*(*Koszt*) zostaje przypisana zmiennej *MinKoszt*).

```
min_koszt(Towar, MinKoszt) ←
        agg (
                towary_podst(Towar, _, Koszt)
                grouped_by [Towar]
                where [MinKoszt isagg min(Koszt)] ).
```

Predykaty rekurencyjne

Dedukcyjne bazy danych, podobnie jak języki programowania w logice, umożliwiają rekurencyjne definiowanie predykatów wirtualnych. W poniższym przykładzie zdefiniowano predykat *towar_komponent* jako przechodnie domknięcie predykatu kompozycja:

```
towar_komponent (Towar, Komponent) ←
        kompozycja(Towar, Komponent, _).
towar_komponent (Towar, Komponent) ←
        kompozycja(Towar, KomponentPośredni, _) and
        towar_komponent(TowarPośredni, Komponent).
```

Mechanizm rekurencji może być łączony z funkcjami zagregowanymi. Przykładowo, w celu wyznaczenia całkowitego kosztu określonego towaru złożonego jest konieczna rekurencyjna penetracja jego drzewa kompozycji oraz sumowanie kosztów poszczególnych komponentów, z uwzględnieniem liczby ich wystąpień. Oczywiście, obliczenie kosztu komponentu złożonego wymaga rekurencyjnej penetracji związanego z nim poddrzewa kompozycji. W przypadku, gdy na danym poziomie kompozycji określony komponent występuje wielokrotnie, wywoływany jest predykat zewnętrzny *times*, wyznaczający iloczyn jego dwóch pierwszych argumentów.

```
koszt_całkowity(Towar, KosztCałk) ←
        agg (
                koszt_komponentu(Towar,Komponent,KosztKompon)
                grouped_by [Towar]
                where [KosztCałk isagg sum(KosztKompon)]
              ).

koszt_komponentu(Towar, Towar, Koszt) ←
        min_koszt(Towar, Koszt).
koszt_komponentu(Towar, Komponent, Koszt) ←
        kompozycja(Towar, Komponent, Liczba) and
        koszt_całkowity(Komponent, CałkKosztKompon) and
        times(Liczba, CałkKosztKompon, Koszt).
```

Rozszerzenia proceduralne

Większość systemów dedukcyjnych baz danych oferuje liczne rozszerzenia proceduralne w celu ułatwienia przygotowania

aplikacji [10]. Typowymi rozszerzeniami są pętle typu

foreach ... **do** ...
oraz
while ... **do** ...

znacznie redukujące liczbę **odcięć** (ang. *cut*) i **wymuszonych nawrotów** (ang. *fail*) języka Prolog.

Śladem relacyjnych baz danych, udostępnia się również użytkownikowi specjalne mechanizmy umożliwiające zarządzanie jego transakcjami. Transakcja, traktowana przez użytkownika jako sekwencja kierowanych do bazy danych zapytań, może zostać przez niego **zatwierdzona** (ang. *commit*), **wycofana** (ang. *abort*) jako całość lub wycofana częściowo, do wcześniej zdefiniowanego **punktu bezpiecznego** (ang. *safe point*).

Bardzo ciekawym mechanizmem, proponowanym przez dedukcyjne bazy danych, są tak zwane **uaktualnienia warunkowe** (ang. *conditional updates*). Określone przez użytkownika zmiany są trwale wprowadzane do bazy danych, jeśli związany z nimi warunek jest spełniony. Wyróżnia się dwa typy warunków:
- warunki wartościowane przed próbą wprowadzenia zmian (ang. *pre-condition*),
- warunki wartościowane po wprowadzeniu zmian (ang. *post-condition*).

Oczywiście w drugim przypadku, w zależności od wartości warunku, wstępnie wprowadzone modyfikacje są zatwierdzane lub wycofywane z bazy danych. Przykładem uaktualnienie warunkowego typu *pre-condition* jest zwiększenie kosztu towaru podstawowego T, dostarczanego przez dostawcę D_1, pod warunkiem, że dostawca D_1 nie jest jedynym dostawcą towaru T (innymi słowy, pod warunkiem, że istnieje inny dostawca D_2 tego samego towaru T). Przykładem uaktualnienia warunkowego typu *post-condition* jest zwiększenie kosztu towaru podstawowego T_1 pod warunkiem, że koszt wytwarzanego z niego towaru złożonego T_2 zwiększy się co najwyżej o 10%. W tym przypadku, decydujemy się na wstępne wprowadzenie zwyżki kosztu towaru podstawowego T_1, a następnie obliczamy całkowity koszt towaru T_2. Jeżeli narzucony przez nas warunek jest spełniony, zwyżka kosztów zostaje zatwierdzona, natomiast w przeciwnym razie zwyżka zostaje wycofana.

Innym ciekawym mechanizmem proponowanym przez dedukcyjne bazy danych są tak zwane **uaktualnienia hipotetyczne** (ang. *hypothetical updates*), odpowiadające stosunkowo często zadawanemu w życiu codziennym pytaniu *co by było gdyby...?* Uaktualnienia hipotetyczne polegają na wstępnym (bezwarunkowym) wprowadzeniu zmian do bazy danych, a następnie wycofaniu (również bezwarunkowym) wszystkich wstępnie wprowadzonych zmian. Przykładem uaktualnienia hipotetycznego jest następujące zapytanie kierowane do bazy danych przez szefa pewnej firmy: *Jaki byłby deficyt budżetowy mojej firmy, gdybym podwyższył zarobki wszystkich pracowników o 10 procent?*

Więzy integralności

Jak wspomniano wcześniej, więzy integralności, podobnie jak struktury danych, służą do opisu statycznych własności modelowanego w bazie danych świata rzeczywistego. W dedukcyjnych bazach danych są one definiowane za pomocą reguł logicznych, co istotnie rozszerza zakres możliwości ich specyfikacji w stosunku do relacyjnych baz danych. Transakcja modyfikująca fakty i/lub reguły dedukcyjnej bazy danych może zostać zatwierdzona wyłącznie wtedy, gdy w wyniku swego działania nie naruszyła zdefiniowanych w bazie danych więzów.

Przykładami więzów integralności mogą być następujące formuły:

(1) *forall [Towar, Dostawca, Koszt]:*
towary_podst (Towar, Dostawca, Koszt)
→
zaufani_dostawcy(Dostawca).

(2) *not exists [X]:*
towar_komponent(X, X).

Pierwsza formuła odwołuje się do dwóch predykatów bazowych: *towary_podst* i *zaufani_dostawcy*. Wyraża ona naturalny wymóg, aby dostawcy magazynowanych towarów byli zaufanymi dostawcami. W konsekwencji, żadna z wykonywanych w systemie transakcji nie może trwale wprowadzić nowego towaru (faktu) do predykatu *towary_podst*, jeżeli dostawca towaru nie występuje w zbiorze faktów predykatu *zaufani_dostawcy*. Zauważmy, że nie jest również możliwe usunięcie dostawcy ze zbioru „zaufanych" dostawców, jeżeli jest magazynowany jakikolwiek towar przez niego dostarczony.

Druga formuła odwołuje się do rekurencyjnie zdefiniowanego predykatu wirtualnego *towar_komponent*. Wyraża on również naturalny wymóg, aby żaden z towarów złożonych nie był zarazem swoim bezpośrednim lub pośrednim komponentem. Innymi słowy, formuła zapewnia acykliczność grafu kompozycji towarów złożonych. W praktyce oznacza to, że usuwanie faktów predykatu *towar_komponent*, jest zawsze możliwe. Natomiast modyfikacja lub wstawianie nowych faktów jest możliwe wyłącznie pod warunkiem niewprowadzenia cyklu w powiązaniu towarów złożonych z ich komponentami.

Zauważmy w tym miejscu, że więzy integralności dedukcyjnych baz danych bardzo ułatwiają precyzyjne modelowanie rzeczywistości. Z drugiej jednak strony, mogą przyczynić się do istotnego zmniejszenia efektywności systemu, wynikającego z konieczności odczytu i wywodu bardzo dużej liczby faktów, przed podjęciem decyzji o zatwierdzeniu lub odrzuceniu transakcji. W przypadku, gdy w magazynie są przechowywane towary niezbędne do wytworzenia np. samochodu, liczba faktów predykatu *kompozycja* jest bardzo duża. Wynika to ze skomplikowanej struktury kompozycyjnej samochodu. Zatwierdzenie niepozornej transakcji, uaktualniającej pojedynczy fakt predykatu *towar_komponent*, wiąże się z koniecznością odczytu (często wielokrotnego) większości lub wszystkich faktów predykatu *towar_komponent*.

W celu pełnego zrozumienia specyfiki więzów integralności dedukcyjnych baz danych przedstawimy teraz ich podstawową klasyfikację.

Najbardziej ogólny podział więzów integralności polega na wyróżnieniu grupy więzów natychmiastowych i opóźnionych. **Więzy natychmiastowe** (ang. *immediate*) muszą być spełnione po wykonaniu każdej elementarnej operacji transakcji. Przykładem jest wymaganie, by atrybut *wiek* przyjmował wartości dodatnie. Naruszenie natychmiastowych więzów integralności powoduje zwykle odrzucenie transakcji bezpośrednio po niepoprawnej operacji.

Więzy opóźnione (ang. *deferred*) muszą być spełnione po zakończeniu ostatniej operacji składającej się na transakcję i mogą być przejściowo naruszane w trakcie realizacji transakcji. Przykładem może być wcześniej wspomniany wymóg acykliczności grafu kompozycji wytwarzanych towarów.

Inny sposób podziału więzów integralności polega na wyróżnieniu grupy więzów statycznych i dynamicznych.

Więzy statyczne (ang. *static*) dotyczą pojedynczego stanu modelowej rzeczywistości. Ich przykładami są następujące wymagania:
1) atrybut *płeć* może przyjmować jedną z dwóch wartości: {*męska, żeńska*},
2) każdy pracownik ma swojego przełożonego,
3) całkowita liczba pracowników każdego wydziału nie może przekraczać 100.

Więzy statyczne dzielimy dalej na zagregowane i niezagregowane. **Więzy zagregowane** (ang. *aggregate*) wymagają podczas weryfikacji użycia co najmniej jednej operacji agregującej, takiej jak sumowanie wartości, obliczanie wartości średniej, minimalnej itp. Przykładowo, weryfikacja trzeciego spośród wyżej zestawionych wymagań wiąże się z koniecznością zliczania pracowników poszczególnych wydziałów.

Wśród statycznych **więzów niezagregowanych** (ang. *non-aggregate*) możemy wyróżnić następujące wymagania:
- niepustej wartości (ang. *not null value*) określonych atrybutów predykatu (bazowego),
- unikalnej wartości atrybutu lub grupy atrybutów w ramach określonego predykatu,
- wartości atrybutu z określonej dziedziny, będącej okrojeniem dostępnego standardowego typu danych lub typem wyliczeniowym (tak jak w przykładzie 1. powyżej),
- wartości atrybutu (lub kombinacji atrybutów) występującej jawnie w innym predykacie,
- zależności funkcyjnej określonego atrybutu od wartości innego atrybutu predykatu.

W odróżnieniu od więzów statycznych, więzy **dynamiczne** (ang. *dynamic*) ograniczają możliwe zmiany stanów modelowej rzeczywistości. Na przykład, dopuszczalne są następujące zmiany stanu cywilnego osób płci męskiej: z kawalera na żonatego, z żonatego na rozwiedzionego, z żonatego na wdowca, z wdowca na żonatego i z rozwiedzionego na żonatego. W związku z tym, niedozwolone są na przykład zmiany stanu cywilnego z kawalera na rozwiedzionego lub z kawalera na wdowca. Innym przykładem jest wymaganie, aby zmiana płacy pracowników zawsze polegała na jej zwiększaniu. Więzy dynamiczne, podobnie jak statyczne, możemy podzielić na zagregowane i niezagregowane, w zależności od tego ile faktów musi być przeanalizowanych w trakcie ich weryfikacji.

Realizacja zapytań i ich optymalizacja

Algorytmy wartościowania zapytań kierowanych do dedukcyjnych baz danych powinny charakteryzować się trzema cechami [2]:
- **poprawnością** (ang. *soundness*), polegającą na tym, że wynik realizacji zapytania nie może zawierać niewłaściwych informacji, tj. informacji nie będących odpowiedzią na zadane zapytanie;
- **kompletnością** (ang. *completness*), polegającą na wyszukaniu wszystkich możliwych odpowiedzi, a nie ich części;
- **skończonością** (ang. *termination*), gwarantującą zakończenie procesu poszukiwania rozwiązań (dotyczy to w szczególności predykatów rekurencyjnych).

Algorytmy te mogą być wykonywane w sposób **opóźniony** (ang. *lazy*) lub **przedwczesny** (ang. *eager*). Pierwszy ze sposobów jest charakterystyczny dla zastosowań baz danych, w których zapytania są generowane *ad hoc*. Uruchomienie algorytmu wartościowania zapytań następuje w chwili pojawienia się jawnego żądania, skierowanego do bazy danych przez użytkownika. Drugi sposób jest stosowany wtedy, gdy część zapytań jest znana a priori (np. w systemach rezerwacji biletów) lub znane są najczęściej wykorzystywane predykaty wirtualne. Jedną z dróg przedwczesnego wartościowania zapytań jest tworzenie tymczasowych predykatów bazowych odpowiadających predykatom wirtualnym. Jak pamiętamy, predykaty takie mogą potem zostać zmaterializowane.

Inny podział algorytmów wartościowania zapytań wynika z kolejności stosowania reguł (predykatów wirtualnych) podczas procesu wnioskowania. Typowo, wyróżnia się podejście zstępujące i wstępujące. W **podejściu zstępującym** (ang. *top-down*) wartościowanie rozpoczyna się od zapytania, które jest następnie rozwijane przy użyciu reguł wywodu predykatów wirtualnych, aż do momentu odwołania się do predykatów bazowych. Przy stosowaniu tego typu techniki jest konieczne wykrywanie cykli w procesie wnioskowania, które uniemożliwiałyby spełnienie warunku skończoności. Konieczne jest również wykrywanie pośrednich etapów wnioskowania, których rozwój nie może przynieść żadnych rozwiązań (lub nowych rozwiązań).

W **podejściu wstępującym** (ang. *bottom-up*) kierunek wnioskowania jest odwrotny. Rozpoczyna się od predykatów bazowych, z których informacja (lub jej część) jest niezbędna do wygenerowania odpowiedzi na zadane zapytanie. Następnie, stosowane są reguły predykatów wirtualnych w celu wywiedzenia nowych faktów (opierając się na faktach, które już wywiedziono). Proces zostaje zakończony z chwilą, gdy stosowanie reguł nie generuje nowych faktów. Zauważmy, że podejście to może charakteryzować się niską efektywnością i dużymi wymaganiami pamięciowymi. Wynika to z możliwości generowania faktów, których nie dotyczy zapytanie, oraz z możliwości wielokrotnego wywodzenia tych samych faktów.

Kolejny podział algorytmów wartościowania zapytań wynika z liczby faktów, które są pobierane z predykatów bazowych w jednym cyklu procesu wnioskowania. Wyróżnia się metody pobierające pojedyncze fakty (ang. *fact-at-a-time*) oraz zbiory faktów (ang. *set-at-a-time*). W metodach pierwszej grupy, po uzyskaniu pierwszego rozwiązania, proces wnioskowania jest wielokrotnie wznawiany, w celu odszukania kolejnych rozwiązań. W metodach drugiej grupy, z predykatów bazowych pobierane są jednorazowo zbiory faktów, które następnie, wraz z rozwojem procesu wnioskowania, są ograniczane w celu eliminacji niewłaściwych rozwiązań.

Każda z metod powyższych grup wymaga na ogół dodatkowych mechanizmów optymalizacji, stosowanych niejako współbieżnie z samą metodą. Zaproponowane metody optymalizacji można podzielić na syntaktyczne i semantyczne.

Metody syntaktyczne (ang. *syntactic*) opierają się na precyzyjnej analizie postaci zapytania oraz reguł wnioskowania, które muszą zostać użyte w celu uzyskania odpowiedzi. Wynikiem tej analizy może być – na przykład – zmiana kolejności wartościowania podzapytań, zmiana kolejności stosowania alternatywnych reguł tego samego predykatu, opóźnienie wartościowania reguł rekurencyjnych itp.

Metody semantyczne (ang. *semantic*) opierają się na dodatkowej wiedzy, która zwykle nie jest jawnie przechowywana w bazie danych, np. wiedzy dotyczącej kierowanych do dedukcyjnej bazy danych zapytań, istniejących więzów integralności, ich wzajemnych powiązań oraz powiązań z predykatami.

Częstym sposobem zwiększenia efektywności systemu jest kompilacja niektórych reguł (lub znanych a priori zapytań) do postaci operacji bezpośrednio odwołujących się do predykatów bazowych. Pojedyncza reguła może mieć wiele przekładów,

generowanych do różnych kombinacji zmiennych wolnych i związanych.

* * *

Na zakończenie przedstawione zostanie krótkie porównanie dedukcyjnych baz danych z systemami programowania w logice, a następnie dedukcyjnych baz danych z relacyjnymi bazami danych.

Dedukcyjne bazy danych a systemy programowania w logice

Struktura klauzul stosowanych w dedukcyjnych bazach danych i systemach programowania w logice jest w ogólności taka sama. Specyfika zastosowań dedukcyjnych baz danych, wymogi efektywnościowe jak również konieczność zapewnienia spójności przyczyniły się jednak do wprowadzenia określonych rozróżnień między bazami dedukcyjnymi a systemami programowania w logice, które przedstawiamy poniżej [2].

- W typowych zastosowaniach dedukcyjnych baz danych liczba przechowywanych faktów jest niewspółmiernie większa od liczby reguł. W systemach programowania w logice proporcje te są na ogół odmienne, często wręcz odwrotne.
- W dedukcyjnych bazach danych zbiór predykatów dzielimy na dwa rozłączne podzbiory: predykatów bazowych i predykatów wirtualnych. Uwzględniając ten podział, każdy fakt w bazie danych jest wystąpieniem jednego z predykatów bazowych, natomiast głowa każdej reguły jest atomem będącym wystąpieniem jednego z predykatów wirtualnych. Innymi słowy, predykat może być opisany albo wyłącznie przez fakty (predykat bazowy), albo wyłącznie przez reguły (predykat wirtualny). Taki podział predykatów ułatwia im reprezentację w pamięci systemu bazy danych i zwiększa efektywność operowania na nich. Oczywiście, systemy programowania w logice nie wprowadzają takiego ograniczenia, co z kolei ułatwia modelowanie encji i powiązań między nimi.
- Język manipulowania danymi dedukcyjnymi baz danych na ogół nie jest wyposażony w funkcje (poza podstawowymi funkcjami arytmetycznymi i agregującymi). Z drugiej strony, stosowanie szerokiego zestawu funkcji jest bardzo powszechne w systemach programowania w logice.
- Klauzule dedukcyjnych baz danych mają ograniczone zakresy (dziedziny). Ograniczeń tego typu nie wprowadza się zwykle w systemach programowania w logice.
- W większości systemów programowania w logice zbiór stałych, funkcji i predykatów jest opisany za pomocą symboli występujących jawnie w programie. Z drugiej strony, język dedukcyjnych baz danych opiera się na predefiniowanym schemacie bazy danych.
- Dedukcyjne bazy danych są wyposażone w wyrafinowane interfejsy użytkowe, oparte na językach wysokiego poziomu, ułatwiające opis i manipulowanie danymi. Interfejsy takie na ogół nie występują lub są bardzo ubogie w przypadku systemów programowania w logice.
- Dedukcyjne bazy danych są wyposażone w mechanizmy definiowania i weryfikacji ograniczeń integralnościowych, które muszą być respektowane przez wyróżnione zbiory danych. Pojęcie ograniczenia integralnościowego nie występuje w systemach programowania w logice.
- W dedukcyjnej bazie danych podstawowy nacisk jest położony na efektywność systemu, podczas gdy w systemie programowania w logice – na jego funkcjonalność. Stąd pewne elementy języków programowania w logice, zwłaszcza te o nieco mniejszym wymiarze praktycznym, nie są oferowane przez dedukcyjne bazy danych. Dotyczy to również składni i semantyki klauzul, które są często rozszerzane w językach programowania w logice.
- Problem efektywności dedukcyjnych baz danych sprowadza się głównie do efektywnej implementacji zarządzania danymi, np. do efektywnego przeszukiwania i odzyskiwania faktów z bazy danych, podczas gdy w systemach programowania w logice efektywność wiąże się głównie z efektywną implementacją mechanizmów wnioskowania.
- W dedukcyjnych bazach danych, w wyniku kierowanego do nich zapytania, odszukiwane są wszystkie możliwe rozwiązania. W konsekwencji, wartościowanie zapytań jest często implementowane za pomocą technik zbiorowych. W systemach programowania w logice często odszukiwane jest pierwsze z możliwych rozwiązań. Stąd, typowa realizacja zapytań opiera się na rekurencyjnym wnioskowaniu i dostępie do pojedynczych faktów.
- Terminologia stosowana w dedukcyjnych bazach danych często różni się od terminologii systemów programowania w logice. W pierwszym przypadku, terminologia jest dziedziczona z relacyjnych systemów baz danych, podczas gdy w drugim przypadku, z logiki matematycznej. Przykładowo, pojęcie *zapytanie* dedukcyjnych baz danych jest odpowiednikami pojęcia *celu* systemów programowania w logice.

Dedukcyjne bazy danych a relacyjne bazy danych

Niekwestionowaną różnicą między relacyjnymi bazami danych jest różnica modelu danych. Model danych dedukcyjnych baz danych jest modelem o rozszerzonej semantyce i podwyższonej ekspresji, co ułatwia modelowanie skomplikowanych fragmentów rzeczywistości. Poniżej zestawiono nieco mniej oczywiste różnice między porównywanymi typami baz danych.

- Predykaty bazowe dedukcyjnych baz danych są odpowiednikami relacji relacyjnych baz danych, a zatem fakty bazowe to jakby krotki. Nie ma podobnego odpowiednika dla predykatów wirtualnych, które umożliwiają wywodzenie nowej informacji na podstawie informacji przechowywanej w sposób jawny. **Perspektywy** (ang. *view*) relacyjnych baz danych jedynie w bardzo zawężonym zakresie mogą pełnić taką rolę.
- Zakres typowych prostych atrybutów predykatów bazowych jest znacznie szerszy niż zakres typów atrybutów relacji. Ponadto, część dedukcyjnych baz danych oferuje konstruktory, umożliwiające agregację atrybutów, np. zbiory, listy itp., które nie są dostępne w relacyjnych bazach danych.
- Predykaty wirtualne dedukcyjnych baz danych mogą być materializowane, co istotnie zwiększa efektywność systemu. Pewnym odpowiednikiem materializacji w dedukcyjnych bazach danych jest grupowanie krotek relacji (ang. *clustering*) w celu np. bardziej efektywnej realizacji tzw. **połączeń** (ang. *join*).
- Więzy integralności dedukcyjnych baz danych, dzięki ich wyrażaniu za pomocą reguł logiki pierwszego rzędu, umożliwiają bardzo precyzyjne modelowanie powiązań między encjami i ograniczeń występujących w modelowanej rzeczywistości. Takich możliwości nie ma w relacyjnych bazach danych, z których większość oferuje co najwyżej weryfikację niepustych i unikalnych wartości atrybutów.
- Relacyjne bazy danych udostępniają ograniczone możliwości dynamicznej modyfikacji bazy danych. W przypadku dedukcyjnych baz danych w zasadzie nie ma ograniczeń w modyfikowaniu reguł, więzów integralności, sygnatur predykatów. Ponadto, w niektórych bazach danych operacje modyfikacji schematu mogą być mieszane z zapytaniami.
- Dedukcyjne bazy danych oferują całkowicie nowe funkcje, takie jak warunkowe uaktualnienia, uaktualnienia hipotetyczne, czy też rekurencyjne predykaty, które istotnie zwiększają zakres zastosowań.

dokończenie na s. 28

ROMAN SIMIŃSKI
Uniwersytet Śląski
Katowice

Projekt i realizacja obiektowo zorientowanego podsystemu komunikacji z użytkownikiem

Współczesny rynek oprogramowania stawia wysokie wymagania przed oprogramowaniem, które miałoby na nim znacząco zaistnieć. Jednym z czynników mających istotny wpływ na atrakcyjność oprogramowania jest łatwość jego obsługi. Wśród systemów z zbliżonych możliwościach, przyszły użytkownik z pewnością faworyzować będzie te, które wyposażone są w wygodny, ergonomiczny i przejrzysty podsystem komunikacji z użytkownikiem (ang. *user interface* – UI). Z racji postępującego w tej dziedzinie procesu standaryzacji (np. SAA/CUA) oraz potencjalnej powielarności elementów podsystemu komunikacji, celowe wydaje się opracowanie i implementacja biblioteki programowej realizującej funkcje komunikacji z użytkownikiem. Tego typu rozwiązania istnieją już dla wielu platform sprzętowych, środowisk systemowych czy implementacji języków programowania. Obecnie zaobserwować można trzy kierunki rozwoju tego typu produktów:

1. Biblioteki dołączane do konkretnych implementacji języków programowania (dla DOS występują np. jako uzupełnienia produktów firm Borland, MicroSoft, Zortech-Symantec).
2. Rozbudowane systemy tworzenia podsystemów komunikacji z użytkownikiem, dedykowane dla całej grupy implementacji języków programowania, dostarczane często z pełnym kodem źródłowym (Vermont Views, Vitamin C, Code Screen itp.).
3. Pakiety dostarczane przez producentów oprogramowania systemowego (lub innych producentów, lecz bazujące na nich) umożliwiające konstruowanie systemów komunikacji z użytkownikiem zgodnie z regułami obowiązującymi w danym środowisku systemowym (SDK dla Windows, OS/2, OSF/Motif, OpenLook).

Dzięki bogactwu propozycji w tej dziedzinie można wybrać rozwiązanie najkorzystniejsze dla określonych potrzeb. Praktyczne doświadczenia jednak dowodzą, że dla dość sporej grupy zastosowań kompleksowe rozwiązania dostarczone przez produkty omówione w pkt. 1, 2 są nieoptymalne.

Istnieją systemy nie wymagające bardzo rozbudowanych UI, stąd zastosowanie narzędzi, takich jak: Turbo Vision czy Vitamin C, powoduje nadmierny wzrost rozmiaru programu, zwiększenie wymagań odnośnie wolnej pamięci itp. Dodatkowo przy tworzeniu UI konieczne jest dostosowanie się do standardów proponowanych przez te pakiety. Mimo reklamowanej łatwości modyfikacji i możliwości dostosowania do indywidualnych potrzeb, proces ten jest trudny i wymaga często dogłębnej znajomości zasad funkcjonowania pakietu. Zbytnie przywiązanie się do takiej biblioteki może też znacznie ograniczyć możliwości przenoszenia kodu źródłowego. Może okazać się również, że postawieni przed koniecznością zmiany środowiska systemowego lub nowej wersji językowej, znajdziemy się w nowych warunkach sam na sam z problemem realizacji podsystemu komunikacji z użytkownikiem (zgodnie z zasłyszanym gdzieś stwierdzeniem – „Turbo Vision i problem z głowy, lecz w głowie nic nie przybyło"). Celowe zatem wydaje się zaprojektowanie i implementacja własnej biblioteki służącej do tworzenia i obsługi UI. Będzie ona uwzględniać nasze własne upodobania, będzie realizować wybrany podzbiór funkcji UI, nie będzie przy tym zbyt rozbudowana, a co za tym idzie będzie umożliwiała sprawną i optymalną implementację.

Nie będzie truizmem stwierdzenie, że użytkownik postrzega aplikację przez pryzmat jej podsystemu komunikacji. Zwykle określone funkcje powiązane są z wybranymi elementami podsystemu. Elementy te nie mają więc charakteru pasywnego. Po uaktywnieniu danego elementu podsystemu komunikacji może on wykonywać określone akcje, dzięki czemu użytkownik podświadomie utożsamia te akcje z danym elementem. Dla użytkownika podsystem komunikacji ma więc charakter obiektowy – dany element UI ma nie tylko swą reprezentację graficzną na ekranie, ale także potrafi wykonać pewne czynności. To intuicyjnie obiektowe postrzeganie podsystemu komunikacji może znaleźć odbicie w jego obiektowo zorientowanym projekcie.

Wybór takiej metodyki jest podyktowany również innymi względami. Jednym z nich jest postulowana elastyczność podsystemu, łatwość jego rozbudowy oraz duża modyfikowalność. Ta ostatnia cecha ma szczególnie istotne znaczenie. Projekt UI nie powinien bowiem odwoływać się do danego zastosowania, jednak powinien umożliwiać późniejszą jego lokalizację w ramach konkretnego zastosowania. Większość tych postulatów można pogodzić stosując konsekwentnie zasady projektowania i programowania obiektowego. Kluczem do elastycznej i efektywnej implementacji będą następujące mechanizmy:

- Abstrakcja i enkapsulacja danych osiągnięta dzięki wykorzystaniu klas;
- Dynamiczne wiązanie akcji związanych z komunikatami przesyłanymi do elementów UI (ang. *dynamic binding, late chaining*) zrealizowane dzięki dziedziczeniu polimorficznemu;
- Rozszerzalność cech funkcjonalnych elementów UI dzięki dziedziczeniu monomorficznemu.

Zastosowanie podejścia obiektowego umożliwia wykorzystanie mechanizmu programowania sterowanego zdarzeniami (ang. *event driven programming*), który stanowi jedną z najpopularniejszych metod sterowania podsystemami komunikacji z użytkownikiem.

Zadania podsystemu komunikacji z użytkownikiem

Jednym z problemów występujących na etapie projektowania podsystemu komunikacji z użytkownikiem jest wybór koncepcji zarządzania elementami tego podsystemu. Dana aplikacja działająca pod kontrolą określonego systemu operacyjnego poddawana jest bowiem różnego rodzaju wymuszeniom, które powodują zwykle zmianę stanu aplikacji. Wymuszenia pochodzące od użytkownika (docierające do systemu za pośrednictwem wybranych urządzeń wejściowych) powodują najczęściej określone zmiany w podsystemie komunikacji, następnie podsystem komunikacji powoduje wykonanie określonych czynności. Podsystem komunikacji spełnia więc następujące funkcje:

1. Jest odpowiedzialny za przepytywanie określonych źródeł wymuszeń. Załóżmy, że zaistnienie pewnego wymuszenia nazwiemy zdarzeniem. Podsystem komunikacji winien każde zdarzenie zidentyfikować i zapamiętać.
2. Zapamiętane zdarzenie powinno być przefiltrowane przez podsystem. Zdarzenia nieistotne, błędne lub docierające ze źródeł przez aplikację nieobsługiwanych powinny zostać w pewien sposób obsłużone przez podsystem. Jednym z możliwych podejść jest ignorowanie takich zdarzeń lub realizacja pewnych specyficznych działań mających np. zapobiec potencjalnym sytuacjom błędnym lub awaryjnym. Podsystem może również dokonać konwersji zdarzenia jednego typu na zdarzenie innego typu, znanego aplikacji.
3. Po przeprowadzeniu działań z pkt. 2 podsystem powinien przystąpić do przekazania określonego zdarzenia do jądra aplikacji, odpowiedzialnego za właściwą obsługę danego zdarzenia. Zwykle bywa to poprzedzone ewentualną zmianą stanu elementów podsystemu (znajdującą swe odbicie na ekranie).
4. Podsystem powinien uczestniczyć w obsłudze zdarzenia bądź to na zasadzie pasywnego monitoringu, bądź umożliwiać użytkownikowi ingerencję z trwającym aktualnie procesem.

Omówione powyżej funkcje podsystemu komunikacji wymagają:

- Precyzyjnego zdefiniowania klas zdarzeń, które mogą zaistnieć. W ramach danej klasy sprecyzowane winny być możliwe zdarzenia. Rozpoznane zdarzenie powinno być zapamiętane w wybranej strukturze danych, znanej wszystkim komponentom podsystemu komunikacji.
- Scentralizowania podsystemu komunikacji, tzn. wyróżnienia takiego elementu, który będzie realizował przepytywanie możliwych źródeł wymuszeń, będzie je identyfikował, klasyfikował zaliczając do określonej klasy zdarzeń i zapamiętywał w wybranej strukturze danych.
- Wyróżniony element podsystemu zwany dalej zarządcą podsystemu komunikacji (ZPK) musi również realizować proces lokalizacji tego elementu podsystemu, do którego dane zdarzenie jest adresowane, przekazywania zdarzenia do tego elementu oraz rozpoznawania faktu obsłużenia (lub nie obsłużenia) zdarzenia.
- Niezależnie od postulowanego scentralizowania podsystemu, poszczególne jego elementy muszą mieć możliwość komunikowania się między sobą bez pośrednictwa ZPK. Jest to konieczne np. w przypadku, gdy otrzymanie zdarzenia przez pewien element powinno pociągnąć za sobą zmianę stanu innych elementów podsystemu w sposób automatyczny, bez udziału ZPK. Wymaga to sprecyzowania mechanizmu komunikacji (wymiany informacji) pomiędzy komponentami podsystemu.

Kolejka zdarzeń

Zdarzenia mogą docierać do aplikacji w dowolnej chwili, nawet wtedy gdy aplikacja nie może obsłużyć nadchodzącego zdarzenia. Aby żadne zdarzenie nie zostało zagubione, korzystne wydaje się wprowadzenie kolejkowania zdarzeń. ZPK po zidentyfikowaniu zdarzenia będzie wstawiać je do globalnej (dla aplikacji) kolejki zdarzeń. Proces przepytywania źródeł zdarzeń będzie polegał na pobieraniu zdarzeń oczekujących w kolejce. Taka globalna kolejka umożliwia również komunikację między elementami podsystemu. Załóżmy, że pewna klasa elementów podsystemu może generować zdarzenia i wstawiać je na początek kolejki. Takie zdarzenie zostanie następnie pobrane do obsługi w pierwszej kolejności. Mechanizm ten można wykorzystać do organizacji obsługi tzw. menu, przycisków (ang. *buttons*), pasków narzędziowych (ang. *tool bar, speed bar*).

Ogólna koncepcja organizacji podsystemu komunikacji z użytkownikiem

Aby pogodzić przedstawione powyżej wymagania i przedstawić jeden z możliwych sposobów organizacji podsystemu, można wykorzystać postulowane wcześniej obiektowe podejście do omawianego problemu. Przyjrzyjmy się elementom wcho-

```
class POINT
{
  public:

    // Konstruktor
    POINT( int x, int y )
    {
      POINT::x = x;
      POINT::y = y;
    }

    // Pobranie współrzędnych punktu
    int   get_x( void ) { return x; }
    int   get_y( void ) { return y; }

    // Ustawienie współrzędnych istniejącego punktu
    void set_point( int x, int y ) { POINT::x = x; POINT::y = y; }
    void set_x( int x ) { POINT::x = x ; }
    void set_y( int y ) { POINT::y = y ; }

  private:
    int x,y;
};
```

Wydruk 1. Klasa reprezentująca punkt ekranu

dzącym zwykle w skład podsystemu komunikacji z użytkownikiem, dokonując klasyfikacji wspólnych cech tych elementów. Najczęściej elementy te są postrzegane przez użytkownika jako pewne wycinki ekranu. Zwykle najlepszym przybliżeniem takiego wycinka bywa prostokąt (chociaż w podsystemach graficznych nie jest to zasadą). Taki prostokąt opisują jednoznacznie współrzędne elementu, które można przedstawić w postaci punktów opisujących np. przeciwległe narożniki.

Załóżmy, że dla roboczego opisu przedstawionych koncepcji używać będziemy konwencji zapisu klas z języka C_{++}. Dla prostoty i przejrzystości stosowane będą znaczne uproszczenia, których w docelowej implementacji być nie powinno. Dotyczą one głównie ograniczenia funkcjonalności definiowanych klas, co ma na celu wyeksponowanie cech najistotniejszych na danym etapie rozważań, jak również pewnych nieścisłości syntaktycznych.

```
class RECTANGLE
{
  public:

    // Konstruktory
    RECTANGLE( int left, int top, int right, int bottom )
    : lt( left, top ), rb( right, bottom )
    {
    }

    // Pobranie parametrów prostokąta
    int get_width ( void ) { return ( rb.get_x()  - lt.get_x() + 1 ); }
    int get_height( void ) { return ( rb.get_y()  - lt.get_y() + 1 ); }

    int get_left  ( void ) { return lt.get_x(); }
    int get_right ( void ) { return rb.get_x(); }
    int get_top   ( void ) { return lt.get_y(); }
    int get_bottom( void ) { return rb.get_y(); }

    // Ustawianie współrzędnych istniejącego prostokąta
    void set_rect( int left, int top, int right, int bottom )
    {
      lt.set_point( left, top );
      rb.set_point( right, bottom );
    }

  protected:
    // Narożniki - lewy górny, prawy dolny
    POINT lt, rb;
};
```

Wydruk 2. Klasa opisująca prostokąt ekranu

```
class ABSTRACT_FORM : public RECTANGLE
// Abstrakcyjna forma ekranowa
{
  public:

  ABSTRACT_FORM( const RECTANGLE & scr_coords ) : RECTANGLE( scr_coords )
  {
  }

  ABSTRACT_FORM( int left, int top, int right, int bottom )

                : RECTANGLE( left, top, right, bottom )

  {
  }

  virtual ~ABSTRACT_FORM( void )
  {
  }

  // Metody wirtualne wymagające przesłonięcia w klasach pochodnych
  virtual int show( void ) = 0;
  virtual int select  ( void ) = 0;
  virtual int unselect( void ) = 0;
  virtual int process_event( EVENT & ) = 0;
  virtual int hide( void ) = 0;
};
```

Wydruk 3. Deklaracja abstrakcyjnej klasy wzorcowej

Punkt ekranu może być reprezentowany przez klasę o definicji przedstawionej na wydruku 1. Wybrany prostokąt ekranu opisywać może klasa przedstawiona na wydruku 2.

Załóżmy, że dowolny element podsystemu będzie miał wszystkie cechy prostokąta. Jednak to zbyt mało. Każdy element powinien być również wyposażony w pewien zestaw standardowych operacji, które może wykonywać. Zastanówmy się, jakie operacje powinien wykonywać dany element podsystemu. Na pewno powinien mieć umiejętność wyświetlania na ekranie swej graficznej reprezentacji. Niech ta operacja nazywa się *show*. Komplementarną operacją będzie wymazanie własnego obrazu z ekranu – tę operację nazwiemy *hide*. Ponieważ w danym momencie na ekranie współistnieć może wiele elementów, musi istnieć mechanizm uaktywniania jednego wybranego elementu. Towarzyszyć temu będzie określone wyróżnienie graficznej reprezentacji danego elementu. Tę funkcję realizować będzie operacja *select* oraz operacja przeciwna *unselect*.

Każdy element podsystemu komunikacji z użytkownikiem powinien potrafić odpowiadać na przesyłane do niego komunikaty, którymi będą (upraszczając sprawę) najczęściej zdarzenia. Zdarzenie przesłane do elementu może zostać obsłużone lub nie a fakt ten powinien zostać zasygnalizowany. Niech operacja obsługi nadesłanego zdarzenia nazywa się *process_event*. Ten zestaw operacji stanowi teoretyczne zadowalające minimum. Wynikiem powyższej analizy może być abstrakcyjna klasa stanowiąca wzorzec, szablon dla dowolnego elementu podsystemu. Z tej klasy wywieść można na zasadzie dziedziczenia dowolny element, dodając nowe metody oraz przesłaniając istniejące już metody wirtualne. Przykładowa deklaracja tej abstrakcyjnej klasy może wyglądać jak na wydruku 3.

Powyższa klasa zawiera deklaracje metod wirtualnych, które muszą być przesłonięte wirtualnie w klasach pochodnych.

Załóżmy, że każdy element podsystemu komunikacji to obiekt klasy pochodnej od *ABSTRACT_FORM*. Zgodnie z zasadami programowania obiektowego, zmiennej wskaźnikowej związanej z typem wskazywanym w postaci klasy bazowej można przypisać wskazanie dowolnego obiektu klasy pochodnej. Załóżmy następujące deklaracje:

```
class SCR_FORM : public
                    ABSTRACT_FORM
// Pewna pochodna klasa zawierająca
// deklaracje własnych metod
// wirtualnych
{
   virtual int show();
      .
      .
      .
   virtual int hide();
};

ABSTRACT_FORM * form_ptr;

// Obiekt klasy SCR_FORM
SCR_FORM  screen_form;
```

Można zatem:

```
form_ptr = & screen_form;
```

oraz:

```
form_ptr->show();
form_ptr->unselect();
.
.
.
form_ptr->select();
.
.
.
form_ptr->hide();
.
.
```

Uaktywnianie metod wirtualnych odbywa się na podstawie typu wskazywanego, a nie typu wskaźnika. Zatem każde z powyższych wywołań spowoduje wykonanie akcji określonych metodami klasy *SCR_FORM*. Jeżeli metoda *show* realizuje wyświetlenie okna dialogowego, to ta operacja zostanie wykonana. Jeżeli metoda *show* powoduje wyświetlenie okna menu, pojawi się ono na ekranie. Uogólniając, wywołanie tej metody spowoduje wyświetlenie takiej graficznej reprezentacji elementu, która została określona w ciele metody *show* tej klasy, którą reprezentuje aktualnie wskazywany obiekt. To samo dotyczy innych metod wirtualnych. Zarządca podsystemu komunikacji może więc manipulować obiektami dowolnej klasy pochodzącej od klasy *ABSTRACT_FORM*. Typowa operacja wyświetlenia na ekranie określonego elementu podsystemu będzie mogła być realizowana zgodnie z umieszczonym powyżej schematem. Efekt tej operacji zależy od natury danego elementu, która nie jest znana zarządcy. Aby takie podejście mogło zostać zrealizowane, każdy element podsystemu musi być obiektem klasy pochodzącej od *ABSTRACT_FORM*, musi on przesłaniać odpowiednie metody wirtualne, które realizować będą odpowiednie dla danego elementu akcje. Zarządca podsystemu komunikacji będzie operował na pewnej strukturze danych, której elementami będą obiekty reprezentujące elementy składowe podsystemu. Wszystkie operacje wykonywane będą za pośrednictwem zmiennych wskaźnikowych, co zapewni uaktywnienie właściwych metod wirtualnych. Przed bliższym przedstawieniem zasady działania zarządcy podsystemu komunikacji należy kilka słów powiedzieć o reprezentacji zdarzeń.

Reprezentacja zdarzeń

Metoda *process_event* otrzymuje jako parametr daną typu *EVENT* – zawierającą opis zaistniałego zdarzenia. Jak może wyglądać deklaracja takiej struktury danych?

Załóżmy, że źródłami zdarzeń mogą być klawiatura i mysza. Musimy zakodować te dwa rodzaje zdarzeń, służyć do tego będzie pole *kind*. Zdarzenie pochodzące z klawiatury jest jednoznacznie identyfikowane przez kod klawisza, zgodny z pewną ustaloną konwencją. Zdarzenie pochodzące od manipulatora, jakim jest mysz, wymaga nieco więcej danych. Powinno być opisane przez pole typu *POINT* zawierające ekranowe współrzędne kursora myszy w chwili zaistnienia zdarzenia. Niech to pole nazywa się *where*. Istotny będzie również odpowiednio zakodowany stan przycisków myszy. Przykładowa uproszczona deklaracja klasy *EVENT* może wyglądać jak na wydruku 4.

Metoda *clear* wywołana dla obiektu klasy *EVENT* powoduje, że pole *kind* otrzymuje wartość *NO_EVENTS*. Metodę tę uaktywniać będą odpowiednie metody *process_event* dla zaznaczenia faktu, iż dane zdarzenie zostało obsłużone. Metody

pool_kbd oraz *pool_mouse* realizują proces przepytywania tych dwóch źródeł zdarzeń oraz zapis wykrytego zdarzenia do kolejki.

```
enum EVENT_KINDS
{
   NO_EVENTS     = 0x0000u,
   FROM_MOUSE    = 0x0001u,
   FROM_KBD      = 0x0002u,
};// EVENT_KINDS

enum MOUSE_EVENTS
{
   MOUSE_LEFT_DOWN     = 0x0001u,
   MOUSE_LEFT_UP       = 0x0002u,
   MOUSE_RIGHT_DOWN    = 0x0004u,
   MOUSE_RIGHT_UP      = 0x0008u,
   MOUSE_MOVED         = 0x0010u,
   MOUSE_REPEAT_DOWN   = 0x0020u
};//MOUSE_EVENTS//

enum KEY_CODES
{
   .
   .
   .
};

class EVENT
{
  public:

   EVENT_KINDS   kind;
   union
   {
     KEY_CODES     key;
     // Kod klawisza
     MOUSE_EVENTS  mouse;
     // Kod zdarzenia MOUSE_EVENTS
   };

  POINT    where;
  int  clear()
  {
     return ( kind = NO_EVENTS );
  }
  int  pool_mouse();
  int  pool_kbd();
};
```

Wydruk 4

Przedstawione wyżej deklaracje klas umożliwiają omówienie zasady działania zarządcy podsystemu. Realizuje on cykliczne przepytywania źródeł zdarzeń. Niech realizuje to funkcja:

look_for_events(*EVENT & e*)

pobierająca zdarzenia z kolejki zdarzeń.

Ogólny algorytm działania zarządcy podsystemu może wyglądać następująco:

```
powtarzaj
{
  pobierz_zdarzenie();
  obsłuż_zdarzenie();
}
dopóki( nie trzeba zakończyć
                    działania )
```

co można zapisać zgodnie ze stosowaną konwencją:

```
EVENT event;
event.kind = NO_EVENTS;

exit_flag = FALSE;
do
{
   look_for_event( event );
   process_event( event );
}
while( ! exit_flag )
```

Oprócz powyższej organizacji obsługi zdarzeń, zarządca podsystemu ma także własną metodę *process_event*, jest więc on szczególnym przypadkiem abstrakcyjnej klasy opisującej element podsystemu komunikacji. Metoda *process_event* realizuje proces obsługi zdarzeń. Stanowi centralne ogniwo podsystemu. Tutaj właśnie następuje zidentyfikowanie elementu, do którego adresowane jest dane zdarzenie oraz inicjowana jest obsługa zdarzenia. Aby zarządca zdarzeń mógł odnaleźć odpowiedni element, muszą one być połączone w pewien łańcuch. Skład tego łańcucha może się zmieniać, istotne jest to, aby w danym momencie zarządca podsystemu mógł w nim odnaleźć właściwy element. Gdy taki istnieje, uaktywniana jest metoda *process_event* tego elementu, odpowiedzialna za właściwą obsługę zdarzenia.

Metoda *process_event* zarządcy podsystemu dokonywać może wstępnego filtrowania zdarzeń, np. zdarzenia związane z uaktywnianiem kolejnych elementów podsystemu będą obsługiwane przez zarządcę, który uaktywniać będzie metody *select* i *unselect* odpowiednich obiektów. Dzięki temu, że wszystkie elementy podsystemu mają tego samego przodka w postaci omówionej klasy abstrakcyjnej, zarządca zdarzeń potrafi nimi manipulować, nie wiedząc nic o ich właściwym wcieleniu. Elementami łańcucha mogą być okna dialogowe, edycyjne, komunikacyjne, elementy menu, przyciski itp. Jeżeli wszystkie są wywiedzione z jednej klasy bazowej oraz konsekwentnie przesłonięte zostały wszystkie metody wirtualne abstrakcyjnej klasy bazowej, działanie zarządcy zdarzeń jest procesem nietrudnym do realizacji. Załóżmy, że łańcuch elementów to dwukierunkowa lista obiektów pewnej klasy wywiedzionej z abstrakcyjnej klasy bazowej (patrz wydruk 5).

Pola *next, prev* stanowią łączniki w liście dwukierunkowej obiektów klasy *SCR_FORM*. Pole *on_screen* określa czy dany element jest wyświetlony na ekranie, pole *selected* mówi, czy dany element jest aktualnie wyróżniony (uaktywniony).

Zarządca podsystemu komunikacji powinien być właścicielem listy obiektów klasy *SCR_FORM*. Powinien on umożliwiać dodanie (usunięcie) elementu do (z) listy. Zwykle jeden element podsystemu jest wyróżniony. Zarządca powinien wiedzieć, który to element. Powinien być wyposażony w możliwości odnajdywania tego elementu, do którego adresowane jest dane zdarzenie. W rzeczywistych podsystemach proces ten bywa bardzo skomplikowany, związane to jest m.in. z problemem obsługi tzw. klawiszy szybkiego reagowania (ang. *hot keys*), pasków narzędziowych itp. Przykładowa definicja klasy reprezentującej zarządcę podsystemu operującego na łańcuchu obiektów przedstawiona jest na wydruku 6.

Do listy można dodawać (i kasować) elementy wykorzystując przeciążone operatory $+=$, $-=$, oraz metody *add* i *sub*. Metody *sel, sel_next, sel_prev* służą zarządcy do uaktywniania aktualnego elementu podsystemu, czemu towarzyszy wywoła-

nie odpowiednich metod *select* i *unselect* dla obiektów stanowiących listę. Metoda *find_who_has_mouse* umożliwia odnalezienie w łańcuchu elementu wskazanego przez kursor myszy. Metoda *exec* realizuje opisany wyżej iteracyjny proces obsługi zdarzeń. Pole *exit_flag* decyduje o zakończeniu działania zarządcy podsystemu. Przykładowa sekwencja programowa obrazująca tworzenie łańcucha elementów podsystemu komunikacji może wyglądać następująco:

```
class DIALOG_WIN : public SCR_FORM
{
  ...
};

class BUTTON : public SCR_FORM
{
  ...
};

class MENU_WIN : public DIALOG_WIN
{
  ...
};
```

```
class EDIT_WIN : public DIALOG_WIN
{
  ...
};

// Deklaracja elementów składowych
// podsystemu

CHAIN   manager;

DIALOG_WIN d;
BUTTON     b;
EDIT_WIN   e;
MENU_WIN   m;

// Budowa struktury podsystemu
manager += d;
manager += b;
manager += e;
manager += m;

// Uruchomienie zarządcy podsystemu
// - uaktywnienie podsystemu

manager.exec();
```

```
class SCR_FORM : public ABSTRACT_FORM
{
  public:

    SCR_FORM( int left, int top, int right, int bottom )
    : ABSTRACT_FORM( left, top, right, bottom ),

    {
      on_screen = selected = FALSE;
      next = prev = NULL;
    }

    virtual ~SCR_FORM( void )
    {
      on_screen = selected = FALSE;
      next = prev = NULL;
    }


    virtual int show( void ) { return ( on_screen = TRUE  ); }
    virtual int hide( void ) { return ( on_screen = FALSE ); }

    virtual int select  ( void ) {  return ( selected = TRUE  ); }
    virtual int unselect( void ) {  return ( selected = FALSE ); }

    virtual int process_event( EVENT & ) { return FALSE; }

  protected:

    int selected;
    int on_screen;

    SCR_FORM * next;
    SCR_FORM * prev;

};
```

Wydruk 5. Deklaracja klasy obiektów będących elementami listy zdarzeń

```
class CHAIN : public SCR_FORM
{
  public:

    CHAIN( int left, int top, int right, int bottom );

    virtual ~CHAIN();

    virtual int show( void );
    virtual int exec( void );
    virtual int run( void );
    virtual int hide( void );
    virtual int select( void );
    virtual int unselect( void );
    virtual int process_event( EVENT & );
    virtual int look_for_event( EVENT & event );

    void operator += ( SCR_FORM & f );
    void operator -= ( SCR_FORM & f );

    add( SCR_FORM * );
    sub( SCR_FORM * );

    sel( SCR_FORM * );
    sel_next( void );
    sel_prev( void );

  protected:
    int      exit_flag;

    // Pierwszy element łańcucha
    SCR_FORM * first;

    // Ostatni element łańcucha
    SCR_FORM * last;

    // Aktualnie wyróżniony element
    SCR_FORM * curr;

    SCR_FORM * find_who_has_mouse( POINT & p );
};
```

Wydruk 6. Definicja klasy reprezentującej zarządcę podsystemu

Okazuje się, że klasa *CHAIN* może służyć nie tylko do globalnego koordynowania działania podsystemu, lecz również może mieć zastosowanie lokalne. Obiekt klasy *CHAIN* może być bowiem elementem łańcucha! Wykorzystując tę cechę można konstruować rozbudowane podsystemy komunikacji, zawierające wiele bardzo wyrafinowanych elementów. Wyobraźmy sobie, że jednym z elementów podsystemu jest pasek narzędziowy złożony z przycisków. Jest więc on łańcuchem elementów. Deklaracja takiego paska dla edytora może wyglądać następująco:

```
CHAIN  tool_bar;
BUTTON    center,    left_just,
right_just, bold, underline;

tool_bar += center;
tool_bar += left_just;
tool_bar += right_just;
tool_bar += bold;
tool_bar += underline;

manager += tool_bar;
```

Obiekt *tool_bar* jest reprezentowany przez jego elementy składowe. Przekazanie zdarzenia do metody *process_event* obiektu *tool_bar* powoduje wykonanie czynności właściwych dla zarządcy podsystemu reprezentowanego przez łańcuch przycisków. Wywołanie metody *exec* spowoduje ograniczenie działania całego podsystemu, aktualnym odbiorcą zdarzeń będzie właśnie *tool_bar* i ten stan będzie występował tak długo aż nie zaistnieją warunki przerywające działanie metody *exec*.

Podsumowując, zarządzanie elementami łańcucha jest stosunkowo mało skomplikowane dzięki zastosowaniu techniki projektowania obiektowego. Wykorzystujemy maksymalnie istniejące już komponenty, mamy możliwość rozszerzania ich własności funkcjonalnych (dziedziczenie, dodawanie metod), jak również ich zmianę bez ingerencji w kod źródłowy (przesłanie metod wirtualnych). Autor zrealizował bibliotekę programową realizującą funkcję podsystemu komunikacji opartą na zasygnalizowanej koncepcji. Współpracuje ona z kompilatorami języka C_{++} dla systemu DOS. Niestety, dokładniejszy opis problemów związanych z tą realizacją oraz możliwych ich rozwiązań przekracza ramy tej publikacji.

Autor zobowiązuje się udzielić wszelkich dodatkowych informacji osobom, którym powyższy opis wydał się zbyt ubogi, mało konkretny lub mało komunikatywny.

Wszystkich zainteresowanych prosimy o kontakt z redakcją, telefon w Warszawie 39-14-34, która umożliwi skomunikowanie się z Autorem artykułu.

GRAŻYNA HOŁODNIK-JANCZURA
ELŻBIETA PŁONKA-SZYDLAK
EWA ZABAWSKA
Instytut Organizacji i Zarządzania
Politechnika Wrocławska

Realizacja przedsięwzięć informatycznych z wykorzystaniem metodyki CASE firmy Oracle

CASE (*Computer-Aided Systems Engineering*), czyli komputerowo wspomagana inżynieria systemów, reprezentuje strukturalne podejście do analizy systemów informacyjnych oraz konstruowania odpowiadających im systemów informatycznych oferując zestaw narzędzi automatyzujących proces ich modelowania i budowy. Narzędzia te obejmują bazę danych dla konstruktorów systemów oraz zbiór programów uwzględniających graficzny interfejs. W zależności od rodzaju narzędzia (zastosowanego pakietu CASE) stopień automatyzacji tego procesu jest zróżnicowany [7, 8].

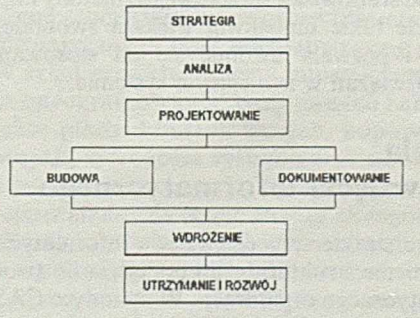
Pakiet CASE firmy Oracle zawiera kompletny zestaw narzędzi wspomagających proces modelowania i automatyzacji systemów informacyjnych dla dużych organizacji. Wykorzystanie tych narzędzi rozpoczyna się od momentu wszczęcia wstępnych badań organizacji i może być kontynuowane po wdrożeniu utworzonego za ich pomocą systemu informatycznego w trakcie jego pielęgnacji i rozwoju. Możliwości wykorzystania narzędzi CASE w trakcie realizacji przedsięwzięcia informatycznego, którego efektem końcowym jest **system informatyczny organizacji**, precyzuje zaproponowana przez firmę Oracle metodyka CASE*Method [2]. Metodyka ta formułuje dla analityków systemów, projektantów i programistów pakiet zadań do wykonania oraz określa zestaw metod i technik, które mogą być zastosowane podczas realizacji przedsięwzięcia.

W artykule zostanie omówiony zestaw tych zadań, których wykonanie jest niezbędne dla powodzenia przedsięwzięcia informatycznego oraz będzie wskazana przydatność techniki prototypowania w jego realizacji. Wykorzystanie narzędzi CASE firmy Oracle w realizacji przedsięwzięć informatycznych, problemy związane z zapewnianiem jakości tworzonych z ich pomocą systemów, metody realizacji poszczególnych zadań, jak również problemy związane z zarządzaniem przedsięwzięciem informatycznym zostaną przedstawione w kolejnych artykułach poświęconych produktom CASE firmy Oracle.

Podstawowe założenia metodyki

Realizacji przedsięwzięć informatycznych mających na celu tworzenie systemów informatycznych spełniających potrzeby dużych organizacji gospodarczych nie jest sprawą łatwą i wymaga pokonania wielu trudności z zakresu zarządzania, technologii informatycznej oraz współpracy z użytkownikami przyszłych systemów i personelem odpowiedzialnym za komputeryzację. Uciążliwości te z różnym nasileniem występują w różnych organizacjach i są uzależnione od specyfiki konkretnej organizacji. Najpoważniejsze problemy są jednak związane z uczestnictwem ludzi w realizacji tych przedsięwzięć. Z jednej strony bowiem występują użytkownicy systemów nie umiejący w precyzyjny sposób określić własnych potrzeb, używający często tych samych słów w różnych kontekstach i znaczeniach. Z drugiej zaś – konstruktorzy systemów, którzy niekiedy nie potrafią lub nie chcą słuchać ich przyszłych użytkowników, broniąc punktu widzenia specjalistów przetwarzania danych.

Zasygnalizowane wyżej problemy w komunikacji na linii użytkownik systemu i jego konstruktor mogą być znacznie pomniejszone, a nawet wyeliminowane dzięki wykorzystaniu proponowanego przez firmę Oracle podejścia do realizacji przedsięwzięć informatycznych zgodnych z metodyką CASE*Method. Metodyka ta jest ukierunkowana na aktywny współudział ludzi będących bezpośrednimi i pośrednimi odbiorcami systemów użytkowych, składających się na produkt finalny przedsięwzięcia informatycznego jakim jest system informatyczny organizacji. Opiera się ona na założeniu, że sukces przedsięwzięcia informatycznego jest uzależniony od pełnego zrozumienia potrzeb organizacji oraz otoczenia, w którym działa. Zrozumienie tych potrzeb można osiągać wyłącznie przez wyraźne oddzielenie analizy (co robić?) od projektowania (jak robić?). Założenie to znalazło wyraz w postulowanym przez firmę Oracle cyklu życia przedsięwzięcia informatycznego (rys.).



Etapy cyklu życia przedsięwzięcia informatycznego

Istotnym elementem omawianej metodyki, umożliwiającym współpracę kierownictwa organizacji oraz bezpośrednich użytkowników z wykonawcami jest ich współuczestnictwo w tzw. sesjach weryfikacyjnych (ang. *feedback session*). Sesje te mają na celu przeprowadzanie obustronnych uzgodnień oraz uzyskanie akceptacji prezentowanych wyników i stanowią aktywną formę udziału przedstawicieli organizacji w realizacji przedsięwzięcia na poziomie koncepcyjnym. Idea sesji weryfikacyjnych wynika z podstawowej przesłanki CASE*Method, jaką jest iteracyjny charakter procesów gromadzenia, kontroli i generowania informacji wykorzystywanych do modelowania organizacji i projektowania systemu. Zgodnie z metodyką wyniki kolejnych iteracji

powinny być poddawane weryfikacji, zaś proces ten powinien być prowadzony tak długo, dopóki nie zostaną one w pełni zrozumiane i zaakceptowane przez przyszłych użytkowników. W zależności od etapu realizacji przedsięwzięcia zadaniem sesji weryfikacyjnych jest:

- **na etapie strategii** – zagwarantowanie pełnego zrozumienia i właściwego zdefiniowania strategii organizacji uwzględniającej przewidywane kierunki jej rozwoju oraz zdefiniowania wynikającej z niej strategii informatyzacji; zadanie to jest realizowane z udziałem kluczowych wykonawców i kierownictwa organizacji;
- **na etapie analizy** – zapewnienie uzyskania kompletnych wymagań ze strony użytkowników; stąd zakłada się ich aktywny udział w sesjach weryfikacyjnych;
- **na etapie projektowania** – opracowanie optymalnego projektu uwzględniającego zarówno wymagania w stosunku do przyszłego systemu, jak i ograniczenia w działalności organizacji oraz ograniczenia realizacyjne; wymaga to ścisłej współpracy członków grupy projektowej z kierownictwem organizacji oraz użytkownikami systemu, w trakcie której wyniki prac projektowych są poddawane ocenie przez zainteresowanych nimi przedstawicieli organizacji.

Kolejnym założeniem w prezentowanej metodyce jest konieczność wygenerowania na każdym z etapów określonych standardowych produktów (ang. *Standard Deliverables*). Produkty te są albo końcowymi rezultatami procesu inżynierskiego, związanymi ściśle z danym etapem, albo produktami tworzonymi w celu zaspokojenia potrzeb następnych etapów. W grupie tych produktów wyróżnia się produkty kluczowe (ang. *Key Deliverables*) oraz produkty specyficzne dla konkretnego projektu. Produkty kluczowe są ściśle określone w metodyce firmy Oracle i ich wygenerowanie jest niezbędne dla powodzenia realizowanego przedsięwzięcia. Produkty specyficzne są tworzone wówczas, gdy nie jest możliwe osiągnięcie założonych celów, opierając się wyłącznie na produktach kluczowych. Nazwy podstawowych produktów kluczowych wyróżniono w dalszej części artykułu pogrubioną czcionką.

W metodyce CASE firmy Oracle zakłada się całkowitą niezależność podstawowego produktu etapu strategii i analizy (tj. modelu organizacji, dla której jest tworzony system informatyczny) od środowiska sprzętowego i metody implementacyjnej. Podejście takie umożliwia większą swobodę w doborze technologii i pozwala uwolnić się od stosowanych, często błędnych rozwiązań w istniejącym systemie.

Cykl życia przedsięwzięcia informatycznego

Etapowy charakter przedsięwzięcia informatycznego ma na celu zapewnienie strukturalnego podejścia do tworzenia systemu informatycznego organizacji. W metodyce CASE*Method, w cyklu życia przedsięwzięcia informatycznego wyróżnia się siedem podstawowych etapów (por. rys.):

- etap strategii, który wiąże się z opracowaniem strategii informatyzacji danej organizacji,
- etap analizy, którego zadaniem jest zdefiniowanie kompletnego zestawu wymagań dla opracowywanego systemu informatycznego,
- etap projektowania, kończący się przygotowaniem projektu przyszłego systemu,
- etap budowy, na którym jest wykonywane oprogramowanie systemu,
- etap dokumentowania, mający zadanie sfinalizowania opracowania pełnej dokumentacji tworzonego systemu,
- etap wdrożenia, na którym następuje testowanie utworzonego systemu w warunkach rzeczywistych,
- etap utrzymania i rozwoju, polegający na śledzeniu pracy systemu, zapewnieniu jego bezawaryjnego działania oraz wprowadzeniu bieżących i planowanych zmian.

Zadania realizowane w poszczególnych etapach oraz generowane w ich ramach podstawowe produkty omówiono poniżej.

Etap strategii

Podstawowym celem etapu strategii jest wypracowanie wraz z kierownictwem organizacji strategii informatyzacji tej organizacji. Przy jej opracowywaniu uwzględnia się zarówno strategię organizacji, w tym jej cele i kierunki rozwoju, jak i funkcje organizacji oraz jej potrzeby informacyjne. Istotnym elementem z punktu widzenia strategii informatyzacji jest ustalenie priorytetów realizacyjnych, będących następstwem przyjętej strategii organizacji. Opracowanie strategii informatyzacji wymaga uwzględnienia również takich czynników, jak wszelkiego rodzaju zagrożenia w osiąganiu celów organizacji oraz ograniczenia w jej funkcjonowaniu.

Podstawowymi produktami otrzymywanymi w wyniku prac realizowanych na etapie strategii są: **ogólny model organizacji, plan informatyzacji** oraz **wstępny projekt architektury systemu**. Produkty te nie mogłyby być uzyskane, bez ustalenia w początkowej fazie etapu strategii, wzajemnych zależności i sposobu współpracy między uczestnikami przedsięwzięcia; w dalszej zaś kolejności, bez pozyskania zgodności co do kierunków rozwoju organizacji. Tak zainicjowana praca powinna również uwzględniać uzgodnienie kompetencji wszystkich uczestników przedsięwzięcia (zarówno bezpośrednich realizatorów, jak i przyszłych użytkowników) w poszczególnych jego etapach.

Przygotowywany na etapie strategii **ogólny model organizacji** powinien odzwierciedlać podstawowe funkcje organizacji oraz jej potrzeby informacyjne. Często, ze względu na złożoność problemów, istnieje konieczność odwzorowania w tym modelu dynamiki przepływu danych, powiązań między funkcjami i potrzebami organizacji, a także powiązań między realizowanymi funkcjami a jednostkami organizacyjnymi wchodzącymi w skład danej organizacji. Jest oczywistym, iż doskonałą podstawą do opracowania tak rozumianego modelu byłaby szczegółowa analiza organizacji. Takie działanie byłoby jednak zbyt kosztowne. W omawianej metodyce zaleca się zatem przeprowadzenie wstępnej analizy, dostarczającej całościowego obrazu organizacji bez koncentrowania się na szczegółach. Jednocześnie, w celu zabezpieczenia się przed dezaktualizacją danych, proponuje się, aby była ona wynikiem częstych, lecz krótkich spotkań z wytypowanymi do współpracy przedstawicielami organizacji. Opracowany w ten sposób ogólny model organizacji powinien uzyskać pełną akceptację przez wszystkich zainteresowanych, zaś sesje weryfikacyjne powinny być wykorzystane do pozyskania całkowitej zgody co do ustalonych w ramach prowadzonych badań bieżących potrzeb organizacji oraz możliwych kierunków jej rozwoju.

Na przygotowywany podczas etapu strategii **plan informatyzacji** składają się: **ramowy plan informatyzacji** oraz **harmonogram realizacji przedsięwzięcia**.

Ramowy plan informatyzacji precyzuje przedmiot, zakres i podstawowe funkcje systemu, priorytety realizacyjne, czas oraz szacunkowe koszty realizacji poszczególnych etapów. Obejmuje propozycje zmian w systemie informacyjnym organizacji i strukturze zatrudnienia oraz wymagania w zakresie szkolenia przyszłych użytkowników systemu. Definiuje również sposób kontroli przebiegu prac oraz zasady współpracy z użyt-

kownikami ze wskazaniem kompetencji w zakresie podejmowania decyzji.

Na etapie strategii, w ramach **harmonogramu realizacji przedsięwzięcia**, przygotowuje się harmonogram prac związanych z analizą poszczególnych części systemu informacyjnego wchodzących w skład tworzonego systemu informatycznego organizacji. Przy jego opracowywaniu uwzględnia się ustaloną wcześniej kolejność wykonywania poszczególnych zadań (priorytety realizacyjne). Harmonogram ten określa sposób prezentacji i zasady zatwierdzania wyników analizy oraz osoby odpowiedzialne za jej przeprowadzenie. Ujmuje szczegółową specyfikę kosztów, realizacji etapu analizy oraz specyfikację środków niezbędnych do jej przeprowadzenia. Precyzuje ponadto przewidywane na tym etapie zmiany w organizacji.

W miarę postępu prac harmonogram realizacji przedsięwzięcia jest sukcesywnie aktualizowany. W trakcie jego aktualizacji, w kolejnych etapach są tworzone szczegółowe harmonogramy prac realizowanych w ich ramach. Harmonogramy te uwzględniają wyszczególnione wcześniej składniki etapu analizy. Sporządza się je albo na etapie, którego bezpośrednio dotyczą, albo, tak jak to przedstawiono powyżej, z odpowiednim wyprzedzeniem. Przygotowany z wyprzedzeniem harmonogram może być aktualizowany bezpośrednio na danym etapie.

Wstępny program architektury systemu jest przygotowywany na podstawie wyników analizy potrzeb organizacji oraz wstępnej analizy jej struktury informacyjnej oraz struktury technicznej. Przy jego opracowywaniu niezbędne jest dokonanie analizy rynku sprzętu, oprogramowania i sieci komputerowych, jak również rozpoznanie dostępnych na rynku technologii. Utworzony na tej podstawie projekt jest z jednej strony podstawą do szacowania kosztów realizacji przedsięwzięcia, z drugiej zaś stanowi punkt wyjścia do opracowania projektu architektury na etapie projektowania.

Proponowane przez firmę Oracle podejście do opracowywania strategii informatyzacji jest metodyczne, a nie – mechanistyczne. Wymaga ono od konstruktorów systemów dużej elastyczności, gdyż wiąże się z koniecznością zawracania z obranej drogi w dowolnym czasie, w przypadku zaistnienia potrzeby skierowania nieporozumień wynikłych w trakcie jej przygotowywania. Zmieniające się warunki, w jakich działa organizacja (zarówno wewnętrzne jak i zewnętrzne), wpływają na zmianę strategii organizacji. Ważne jest zatem, aby w trakcie opracowywania strategii informatyzacji zostały rozpatrzone alternatywne podejścia do osiągania celów organizacji, uwzględniające istniejące ograniczenia w jej działaniu, priorytety realizacyjne oraz możliwe sposoby rozwiązań.

Etap analizy

Opracowywany na etapie strategii ogólny model organizacji stanowi punkt wyjścia do przeprowadzenia pogłębionej analizy organizacji, zgodnie z przygotowywanym również na tym etapie harmonogramem realizacji przedsięwzięcia. W trakcie etapu analizy następuje uszczegółowienie tego harmonogramu.

Podstawowymi produktami etapu analizy są: **szczegółowy model organizacji** oraz **zaktualizowany plan informatyzacji**.

Szczegółowy model organizacji odwzorowuje organizację w ujęciu statycznym oraz dynamicznym. Ujęcie statyczne odzwierciedlają modele podstawowych struktur organizacji, tj. struktury organizacyjnej, funkcjonalnej, informacyjnej i przestrzennej, jak również powiązania struktur: funkcjonalnej z organizacyjną, organizacyjnej z informacyjną oraz organizacyjnej z przestrzenną. W ujęciu dynamicznym model ten odwzorowuje powiązanie struktury funkcjonalnej ze strukturą informacyjną.

Zgodnie z ogólnymi założeniami metodyki, szczegółowa analiza organizacji jest procesem iteracyjnym. Wymaga ona, w pierwszym kroku, skonfrontowania opracowanej strategii informatyzacji z uściślonymi potrzebami przyszłych użytkowników systemu (określonymi na podstawie wywiadów i ankiet) oraz stanem organizacji (określonym w wyniku analizy jej dokumentacji). W kolejnym kroku, wymaga połączenia tych elementów i zrównoważenia punktu widzenia użytkowników z istniejącym systemem i zdefiniowaną strategią informatyzacji. Następnym składnikiem analizy jest skonkretyzowanie rzeczywistych celów organizacji, szczegółowe zdefiniowanie jej funkcji i danych oraz ustalenie preferowanego przez użytkowników stylu pracy. Przeprowadzona dalsza weryfikacja oraz pogłębiona analiza (uwzględniająca zweryfikowane wcześniej różnego typu ograniczenia w funkcjonowaniu organizacji) pozwalają zdefiniować wymagania dotyczące rozproszonego przetwarzania, określić potrzeby w zakresie kopiowania, archiwowania i odtwarzania danych oraz sformułować strategię wdrożenia. Uwzględnienie strategii wdrożenia na etapie analizy, a więc rozpatrzenie procesu przejścia między starym i nowym systemem pomaga w zidentyfikowaniu przyszłych zmian w organizacji oraz pozwala bardziej precyzyjnie określić plan informatyzacji (stąd konieczność aktualizacji na tym etapie ramowego planu informatyzacji).

Etap analizy, to etap ścisłej współpracy analityków systemu z jego przyszłymi użytkownikami. Współpraca ta ma na celu sprawne ustalenie rzeczywistych potrzeb organizacji oraz zbadanie jej aktualnego stanu. W jej ramach użytkownik powinien mieć możliwość bieżącego kontrolowania prowadzonych badań oraz ingerowania w tych sytuacjach, w których pominięcie istotnych szczegółów mogłoby wpłynąć na zagrożenie realizacji celów organizacji. Od efektywności współdziałania analityków z użytkownikami zależy precyzyjne i szybkie zdefiniowanie kompletnego zestawu wymagań, umożliwiającego rozpatrzenie wielu alternatywnych rozwiązań na etapie projektowania.

Etap projektowania

Podstawowymi produktami etapu projektowania są: **projekt systemu informatycznego** oraz **zaktualizowany plan informatyzacji**. W skład projektu systemu informatycznego wchodzą **projekty: oprogramowania, bazy danych i architektury systemu**, a także **plany: kontroli danych, kopiowania i odtwarzania danych, wdrożenia systemu** oraz **testowania systemu**. Podstawą do opracowania wyróżnionych elementów projektu jest sporządzony na etapie analizy szczegółowy model organizacji. W trakcie prac nad projektem uwzględnia się także wszystkie szczegółowe wymagania, sformułowane na etapie analizy, oraz opracowany na etapie strategii wstępny projekt architektury systemu. Na etapie projektowania są również prowadzone uzgodnienia dotyczące wymaganego poziomu automatyzacji systemu informacyjnego oraz jego struktury technicznej. Końcowy rezultat tych uzgodnień ma bezpośredni wpływ na tworzony na tym etapie projekt architektury systemu oraz oddziaływuje na proces opracowywania całego projektu. Wymagania z poziomu analizy oraz przeprowadzone uzgodnienia dotyczące środowiska sprzętowo-programowego znajdują także odzwierciedlenie w przygotowywanym na początku tego etapu harmonogramie prac, aktualizującym zarówno harmonogram realizacji przedsięwzięcia, jak i ramowy plan informatyzacji.

Proces opracowywania projektu jest w głównej mierze procesem iteracyjnym. W trakcie jego realizacji może nastąpić

modyfikacja wymagań sformułowanych w etapie analizy. Wymagania te są uwzględnione w alternatywnych projektach tworzonych na etapie projektowania. Przygotowywane warianty projektów są następnie poddawane weryfikacji. Weryfikacja tych projektów, a zatem i wymagań, jest prowadzona tak długo, dopóki nie zostanie osiągnięty kompromis w sprawie końcowego rozwiązania. W trakcie procesu projektowania może się jednak okazać, że zrealizowanie stawianych wymagań jest mało realne. Powodem tego może być np. zbyt długi czas realizacji, brak środków finansowych, brak fachowców lub też ograniczenia sprzętowe. Zdarza się również, że pojawia się bariera związana z brakiem narzędzi i technik niezbędnych do realizacji postawionych wymagań. Może się także okazać, że o ile jest możliwe niezależne wygenerowanie projektu oprogramowania, projektu bazy danych oraz projektu architektury systemu, to nie jest realne ich połączenie. W takich przypadkach istnieje konieczność powrotu do etapów analizy i/lub strategii, gdzie musi nastąpić korekta stawianych wymagań i ich dostosowanie do ograniczeń wynikających z założonej technologii i ograniczeń sprzętowych.

Wynikiem tak prowadzonego procesu iteracyjnego jest wstępny projekt systemu obejmujący dopasowane do siebie **projekty: oprogramowania, bazy danych** i **architektury systemu**. W dalszej fazie etapu projektowania, dla tak opracowanego projektu wstępnego są tworzone szczegółowe **plany: kontroli danych, kopiowania i odtwarzania danych, testowania** oraz **plan wdrożenia systemu**, zawierający plan jego przekazywania i odbioru, szkolenia, zmian w organizacji i plan instalacji systemu. Faza opracowywania tych planów, podobnie jak faza tworzenia wstępnego projektu systemu, może być procesem iteracyjnym. Zatwierdzenie prac projektowych przez kierownika projektu oraz osoby odpowiedzialne za współpracę ze strony organizacji kończy budowę projektu systemu informatycznego.

Projekt systemu informatycznego organizacji powinien uwzględniać wszystkie aspekty projektowania oraz wynikające z nich ich wzajemne powiązania. Dlatego też, istotnym zadaniem kierownika projektu jest zapewnienie współpracy członków grupy projektowej i niedopuszczanie do ich rozmyślnej izolacji. Wpływ na realizację całego projektu powinni mieć również przedstawiciele organizacji, tj. użytkownicy, menadżerowie, personel operacyjny i personel odpowiedzialny za eksploatację systemu. Winni oni uzyskać prawo do ingerencji w prace projektowe polegające na podejmowaniu decyzji, kontroli pracy projektantów oraz zatwierdzaniu końcowych rezultatów.

Etap budowy i dokumentowania

Etap budowy jest ściśle związany z kodowaniem programów i tworzeniem baz danych dla zaprojektowanych systemów użytkowych. Proces ten może być realizowany w sposób zautomatyzowany za pomocą oferowanych przez firmę Oracle narzędzi do generowania definicji baz danych i kodu modułów programowych. Stopień automatyzacji etapu budowy jest bezpośrednio uzależniony od uzgodnionego na etapie projektowania środowiska programowo-sprzętowego oraz stopnia skomplikowania programów. Opracowanie programów oraz ich testowanie (zgodnie z przygotowanymi na tym etapie testami) jest poprzedzone sformułowaniem planu realizacji oprogramowania.

Budowa oprogramowania jest realizowana w pewnym odosobnieniu od użytkownika. Ważne jest jednak, aby wszelkiego rodzaju zmiany stanowiące odstępstwo od przyjętych w projekcie założeń, były konsultowane i akceptowane przez użytkowników i/lub analityka. W związku z tym, programiści powinni być zobowiązani do dokumentowania decyzji związanych z tymi odstępstwami. Użytkownicy natomiast powinni mieć sposobność przedyskutowania z programistą zagadnień związanych z ich oczekiwaniami, określonymi w tzw. preferencjach użytkownika (ang. *user preferences*). W ten sposób użytkownicy nie będą zaskakiwani niespodziewanymi decyzjami podejmowanymi w trakcie etapu budowy systemu. Istotne jest również zapewnienie każdemu programiście, w początkowej fazie etapu budowy, wglądu w całość systemu i umiejscowienie w niej tej jego części, którą będzie oprogramowywał, jak również tych elementów systemu (systemów), z których zasobów będzie korzystał. Takie postępowanie zapewni programistom lepsze przygotowanie do podejmowania decyzji na etapie budowy.

Produktami końcowymi etapu budowy są: **działające** i przetestowane pod kątem formalnej poprawności **oprogramowanie** oraz **baza danych systemu**. Produkty te są tworzone na podstawie przygotowanego na etapie projektu systemu informatycznego oraz opracowanego w początkowej fazie etapu budowy planu realizacji oprogramowania.

Przebiegający równolegle z etapem budowy etap dokumentowania dostarcza **podręczną dokumentację obsługi** dla operatorów systemu oraz **kompletną dokumentację użytkową** dla jego przyszłych użytkowników. Obie dokumentacje są końcowymi produktami etapu dokumentowania, a ich wstępny projekt jest tworzony na etapie projektowania. Zarówno dokumentacja obsługi, jak i dokumentacja użytkowa są opracowywane na podstawie zdokumentowanych wcześniej, za pomocą odpowiednich narzędzi CASE, wyników prac etapów strategii, analizy oraz projektowania i powinny być ukończone przed rozpoczęciem testowania systemu. Wymóg ich przygotowania przed przystąpieniem do testowania jest ściśle związany z ich bezpośrednim wykorzystaniem w tym procesie podczas etapu budowy. Podstawowym zadaniem tych dokumentacji na etapach wdrożenia, utrzymania i rozwoju jest wspomaganie w bieżącej pracy z systemem zarówno operatorów systemów, jak i ich użytkowników.

W nowoczesnych systemach wiele funkcji pełnionych przez dokumentację użytkową przejmuje system bieżącej pomocy (*Help*). System ten pełni rolę dokumentacji typu on-line. Oferowane przez firmę Oracle narzędzia CASE umożliwiają tworzenie tego rodzaju dokumentacji. Jej wstępny projekt dla systemów, dla których przewidziano jej opracowanie, powinien być przygotowany w ramach projektu oprogramowania na etapie projektowania. W tym przypadku, oprogramowany na etapie budowy system pomocy bieżącej, równolegle z dokumentacjami obsługi i użytkową wspomaga procesy testowania, wdrażania oraz utrzymania systemu.

Proces testowania systemów użytkowych, poprzedzony przygotowaniem odpowiednich testów, powinien być przeprowadzony na podstawie opracowanego na etapie projektowania planu testowania systemu, ale również oddzielnie dla tych jego części, dla których zostały wprowadzone dodatkowe ograniczenia (np. związane z czasem pracy lub czasem dostępu, ograniczeniem wartości atrybutów, ograniczeniem liczby wyświetlanych pozycji). Testy te należy przeprowadzić przed przystąpieniem do testowania całego systemu. Od jakości testów i ich struktury zależeć będzie bowiem zachowanie opracowanego systemu w docelowym środowisku sprzętowym.

Etap wdrożenia

W metodyce CASE firmy Oracle zakłada się, iż etap wdrożenia przebiega zgodnie ze zaktualizowanym na etapie budowy planem wdrożenia. Opracowane systemy użytkowe są wprowa-

dzane ostrożnie, z zachowaniem pełnej integracji z istniejącym systemem informacyjnym. W procesie wdrażania uczestniczą zarówno przyszli użytkownicy tych systemów, jak i personel odpowiedzialny za ich utrzymanie i obsługę oraz wykonawcy odpowiedzialni za przebieg wdrożenia. Jest to etap testowania systemu w praktyce.

Na etapie wdrożenia odbywa się szkolenie przyszłych użytkowników i personelu operacyjnego, odbiór testów przeprowadzonych na etapie budowy oraz zainstalowanie sprzętu komputerowego i oprogramowania. Po wprowadzeniu rzeczywistych danych i wykonaniu próbnej implementacji jest przeprowadzana analiza pracy systemu na danych rzeczywistych, w wyniku której wprowadza się bieżące korekty i udoskonalenia zgodnie z zapotrzebowaniem użytkowników.

Podczas etapu wdrożenia kierownictwo organizacji może zażyczyć sobie, aby oba systemy, stary i nowy, pracowały równolegle tak długo, aż nie zostaną osiągnięte cele organizacji, nakreślone w planie informatyzacji. Taka sytuacja zdarza się często, zwłaszcza w organizacjach o konserwatywnym nastawieniu. Wydłuża to proces osiągania zaplanowanych efektów, np. przez rozłożenie w czasie zmian w strukturze zatrudnienia, jednakże nie może być powodem negatywnych reakcji ze strony realizatorów, ale wyzwaniem do szybszego przezwyciężenia problemów związanych z wdrożeniem nowego systemu i osiągnięcia zaplanowanych efektów.

Problemy wymagające przezwyciężenia na etapie wdrożenia mogą być liczne. Mogą one np. wiązać się z negatywnym nastawieniem przyszłych użytkowników, zakłóceniami w koordynacji zaplanowanych prac, a także trudnościami z dotrzymaniem założonych terminów. Część z nich może wynikać z braku możliwości zapewnienia ciągłości pracy, trudności organizacyjnych oraz innych nieprzewidzianych kwestii spornych. Dlatego też tak istotne jest bardzo dokładne zaplanowanie tego etapu. Umożliwia to wczesne zidentyfikowanie możliwych przeszkód i rozstrzygnięcie rodzaju działań, jakie należy podjąć w przypadku ich wystąpienia. W omawianej metodyce do planowania etapu wdrożenia przywiązuje się ogromną wagę. Stąd problemy związane z wdrożeniem systemu informatycznego organizacji są rozstrzygane już od etapu strategii.

Zapewnienie ciągłości pracy organizacji w trakcie wdrażania systemów informatycznych jest celem, który można osiągnąć wyłącznie przez współdziałanie zespołu realizatorów z przyszłymi użytkownikami tych systemów. Jak już podkreślano, współpraca ta powinna przebiegać w trakcie realizacji całego przedsięwzięcia. Użytkownicy powinni być ze znacznym wyprzedzeniem zapoznawani z każdą zaplanowaną implementacją. W ten sposób są oni przygotowywani do dokładniejszej kontroli i porządkowania bieżących danych, przeglądu dotychczasowych procedur oraz współpracy z obsługą systemu. Przygotowanie to powinno zaowocować przy pozyskiwaniu danych ze starego systemu i wpłynąć na skrócenie czasu ich przeniesienia, jak również na szybką prezentację możliwości nowego systemu.

Preferowany w omawianej metodyce proces szkolenia użytkowników, to szkolenie dwustopniowe. Pierwsza jego faza dotyczy wybranej grupy użytkowników zwanych użytkownikami kluczowymi. Ich szkolenie rozpoczyna się na etapie budowy i polega początkowo na podnoszeniu ich zdolności manualnych. Na etapie wdrożenia, kluczowi użytkownicy opanowują pracę z systemem na podstawie rzeczywistych danych i poznają tajniki związane z metodami szkolenia. Jednocześnie, zapoznają się oni z zawartością dokumentacji użytkowej oraz możliwościami jej wykorzystania. W drugiej fazie szkolenia, przygotowani w ten sposób użytkownicy kluczowi szkolą pozostałych użytkowników wdrażanego systemu. Przyjęta metoda szkolenia jest uznawana za metodę bardziej efektywną niż szkolenie wszystkich użytkowników przez realizatorów przedsięwzięcia.

Odbiór systemu może nastąpić po jego pełnej akceptacji przez użytkowników, którą można osiągnąć tylko wówczas, gdy użytkownicy w pełni zapoznają się z nowym systemem oraz docenią osiągane dzięki niemu korzyści.

Etap utrzymania i rozwoju

Zgodnie z omawianą metodyką, etap utrzymania i rozwoju powinien gwarantować sprawne funkcjonowanie systemu w warunkach rzeczywistych przy ograniczeniu do minimum interwencji ze strony bezpośredniej obsługi oraz personelu odpowiedzialnego za jego utrzymanie. W trakcie eksploatacji systemu należy kontrolować sposób jego użytkowania, a także jego wydajność (szybkość działania). Niezbędne zmiany powinny być wprowadzane tak, aby nie powodowały przerw w pracy systemu i były w pełni akceptowane przez użytkowników.

W trakcie obsługi systemu użytkownik powinien uzyskać od personelu odpowiedzialnego za jego eksploatację odpowiedzi na nurtujące go problemy i zapytania. Wszelkie uwagi, życzenia i zastrzeżenia związane z pracą systemu wymagają starannego przeanalizowania i uwzględnienia w fazie jego rozwoju.

Jak już podkreślano, zarówno warunki funkcjonowania organizacji, jak i jej cele mogą się zmieniać w czasie powodując konieczność zmian w strategii organizacji. Zmiany te powinny być uwzględniane w systemie informacyjnym organizacji. Mogą one wiązać się z dołączaniem nowych lub rezygnacją z dotychczas realizowanych funkcji, zmianą struktury bazy danych czy też z koniecznością wbudowania w istniejący system informatyczny organizacji nowego systemu użytkowego korzystającego z istniejącej bazy danych. Dzięki narzędziom CASE firmy Oracle zadanie to można zrealizować względnie szybko przy ograniczeniu do minimum zakłóceń w pracy wcześniej zaimplementowanego systemu.

Miejsce techniki prototypowania w metodyce CASE*Method

Za jedną z najbardziej użytecznych technik stosowanych w realizacji przedsięwzięć informatycznych uznaje się w metodyce CASE*Method technikę prototypowania. Jest ona proponowana również w innych metodykach, na podstawie których są tworzone systemy informatyczne organizacji [6], jednakże sposób jej wykorzystania w cyklu życia przedsięwzięcia informatycznego jest uzależniony od możliwości narzędzi zastosowanych w tym cyklu [3]. Wsparcie procesu konstruowania systemów informatycznych narzędziami CASE firmy Oracle pozwala budować prototypy na różnych etapach realizacji przedsięwzięcia, począwszy od etapu strategii. W omawianej metodyce zaleca się jednak w pełni kontrolowane ich stosowanie. Wykorzystanie techniki prototypowania wyłącznie w celu tzw. szybkiej budowy systemu, sprowadzającej się często do tworzenia programów komputerowych opartych wyłącznie na formach ekranowych może doprowadzić do powstania systemów, dla których nie będzie możliwe uzyskanie ich pełnej dokumentacji, a także (szczególnie przy tworzeniu systemów dla dużych organizacji) może spowodować wygenerowanie w ramach jednej organizacji niezależnie działających systemów użytkowych. Wzajemna niezależność systemów użytkowych ma swoje konsekwencje w dublowaniu funkcji i danych, a to z kolei może

wpłynąć na obniżenie efektywności i wiarygodności systemu całej organizacji. W efekcie końcowym, pomimo, że poszczególne systemy użytkowe będą gwarantować osiąganie postawionych przed nimi celów, może okazać się nierealnym osiągnięcie celów całej organizacji. Stąd, w metodyce CASE firmy Oracle, zaleca się stosowanie techniki prototypowania przede wszystkim jako środka pomocniczego do konstruowania systemu oraz w celu analizowania wariantowych rozwiązań na poszczególnych etapach jego realizacji. I tak:

● **Na etapie strategii** zaleca się wykorzystanie dostępnych w pakiecie CASE firmy Oracle narzędzi do budowy modelu organizacji, na który składają się:

– diagram Obiekt–Relacja (w modelu tym wykorzystano podejście ER, ang. *entityrelationship*, do zobrazowania struktury informacyjnej organizacji [1]),

– diagram hierarchii funkcji, ilustrujący strukturę funkcjonalną organizacji,

– diagramy przepływu danych (ang. *data flow diagrams*) opierające się na diagramach Gane'a i Sarsona [5] i odzwierciedlające powiązanie struktury funkcjonalnej organizacji z jej strukturą informacyjną.

Opracowany w ten sposób model organizacji, uzupełniony o utworzone za pomocą narzędzi CASE makiety form ekranowych, pozwala przeanalizować i zweryfikować zarówno stan organizacji, jak i przyszłe potrzeby. Warianty rozwiązań, uwzględniając aktualne i przyszłe potrzeby organizacji, mogą być przetestowane na przykładach przy wykorzystaniu analizy *what-if* oraz za pomocą różnych modeli finansowych. Wyniki przeprowadzonych analiz są podstawą do aktualizacji zweryfikowanego w ten sposób modelu organizacji.

● **Na etapie analizy** proponuje się wykorzystanie techniki prototypowania w celu kontroli kompletności wymagań skierowanych do przyszłego systemu i zbadania ich wzajemnych uwarunkowań (wymagania te są opracowywane w postaci wariantów). Ponadto, istnieje możliwość przeanalizowania, w zależności od rozważanego wariantu strategii organizacji, roli jaką może pełnić organizacja oraz wynikających z tego tytułu zobowiązań. Często korzystne jest rozpatrzenie, dla każdego z analizowanych wariantów, alternatywnych stylów pracy w powiązaniu z różnymi rolami, jakie w uwzględnianych rozwiązaniach mogą pełnić użytkownicy. Zalecane na tym etapie techniki prototypowania opierają się na narzędziach umożliwiających generowanie form ekranowych (ang. *forms*), ekranów (ang. *screens*) i raportów (ang. *reports*) oraz na narzędziach pozwalających budować tablice odzwierciedlające powiązania między elementami poszczególnych struktur zdefiniowanych w modelu organizacji. Istotne jest, aby technika prototypowania nie była stosowana wyłącznie w celu kontroli realizowanych procesów.

● **Na etapie projektowania** prototypowanie jest postrzegane inaczej. Wykorzystanie tej techniki w procesie tworzenia projektu systemu ma na celu przede wszystkim przeanalizowanie wykonalności wymagań zdefiniowanych na etapie analizy. Zadanie to może często okazać się bardzo trudne. Rozpatrzenie poszczególnych rozwiązań i zawyrokowanie, czy dany projekt jest wykonalny i czy zaspokoi wymagania przyszłych użytkowników systemu, może być zrealizowane przez budowę odpowiadających tym rozwiązaniom makiet. Zakłada się, iż każda z opracowanych makiet powinna być zbudowana dla reprezentatywnego podzbioru systemu uwzględniającego wszystkie założone kontrole i realizowane procesy. Dla kompleksowych systemów prototypowanie można rozszerzyć o badanie realności połączeń między zaproponowanymi w projekcie: systemem zarządzania bazą danych, zasadami monitorowania przetwarzanych transakcji, protokołem sieci, językami programowania i sprzętem komputerowym. Takie postępowanie pozwoli podjąć ostateczne decyzje co do możliwości ich wzajemnej współpracy.

● **Podczas etapu budowy** narzędzia do prototypowania mogą być wykorzystane albo do wspomagania przyrostowej budowy systemu, albo do jego częściowej specyfikacji. Narzędzia te wspomagają przyrostową budowę wówczas, gdy mogą generować na wyjściu nową jakość, tj. część składową przyszłego systemu. Natomiast opracowane za ich pomocą specyfikacje podlegają częściowej lub całkowitej przebudowie przy wykorzystaniu innych narzędzi, pozwalających na uzyskanie systemu o wyższej jakości z punktu widzenia jego wydajności oraz przydatności dla końcowych użytkowników.

W metodyce CASE firmy Oracle wyróżnia się pięć podstawowych klas prototypów. Każda z nich ma na wejściu inny bodziec inicjujący budowę prototypu określonej klasy. Wydzielone klasy obejmują prototypy umożliwiające:

● śledzenie szybkości działania – prawdopodobne bodźce: duże zbiory bazy danych, duża częstotliwość wykonywanych funkcji, wymagany krótki czas odpowiedzi;

● testowanie przydatności systemu dla organizacji – prawdopodobne bodźce: nowe technologie, duża liczba użytkowników, użytkownicy uprzywilejowani;

● badanie wykonalności stawianych wymagań – prawdopodobne bodźce: nowe technologie, nowe systemy użytkowe;

● badanie jakości wymagań – prawdopodobne bodźce: dwuznaczne lub kolidujące ze sobą wymagania, potrzeba budowy nowych systemów;

● wspomaganie przyrostowej budowy – prawdopodobne bodźce: typowe operacje przetwarzania danych oraz obsługa interfejsu użytkownika (ekrany do wprowadzania i aktualizacji danych o niskim lub średnim stopniu złożoności, większość raportów, ekrany zapytań wg wzorca *Query-by-example*, menu systemu, procedury konwersji danych).

W metodyce CASE firmy Oracle prototypowanie jest postrzegane jako użyteczna technika, wykorzystywana podczas etapów strategii, analizy, projektowania i budowy, jednakże wyłącznie jako środek wspomagający realizację przedsięwzięć informatycznych, a nie jako sposób uzyskiwania ich końcowego efektu.

LITERATURA

[1] Barker R.: CASE*Method-Entity Relationship Modelling. Addison-Wesley Publishing Company, 1990

[2] Barker R.: CASE*Method-Tasks and Deliverables. Addison-Wesley Publishing Company, 1990

[3] Fuglewicz P.: Systemy CASE – problemy, techniki, rozwiązania, INFORMATYKA nr 11, 12, 1990

[4] Gane C.: Rapid System Development-Using Structured Techniques and Relational Technology, Prentice-Hall, Englewood Cliffs, New Jersey, 1989

[5] Gane C., Sarson T.: Structured Systems Analysis-Tools and Techniques, Prentice-Hall, Englewood Cliffs, New Jersey, 1979

[6] Martin J.: Information Engineering: a trilogy, Prentice-Hall, Englewood Cliffs, New Jersey, 1989

[7] Metodologie tworzenia systemów informatycznych a CASE. Materiały 5. Wiosennej Szkoły PTI, Świnoujście, 1992

[8] Szyjewski Z.: Klasyfikacja narzędzi wspomagania CASE. INFORMATYKA nr 4, 1993.

Czy macie Państwo zamówioną prenumeratę INFORMATYKI na 1994 rok?

II Krajowa Konferencja poświęcona Elektronicznej Wymianie Danych EDI odbędzie się w Arturówku k. Łodzi (dojazd autobusem miejskim z dworca Łódź-Fabryczna) w dniach od 30 maja do 1 czerwca 1994 r.

Przedmiotem obrad będą problemy budowania, wdrażania i eksploatacji systemów EDI oraz przegląd realnych dokonań i doświadczeń w tym zakresie w naszym kraju.

Konferencja jest adresowana do wszystkich obecnych i przyszłych użytkowników systemów EDI, zwłaszcza reprezentujących przedsiębiorstwa przemysłowe, handlowe, bankowe i transportowe zaangażowane w operacje zagraniczne.

Informacji dotyczących konferencji udziela ***prof. dr hab. Marian Niedźwiedziński, 91-844 Łódź, ul. Szklana 2/14 m. 3, tel., faks: (0-42) 55-67-57.***

Poniżej publikujemy wybrane referaty I Konferencji tego cyklu, która odbyła się w ubiegłym roku.

I Krajowa Konferencja Elektronicznej Wymiany Danych
Arturówek k. Łodzi, 15–17 czerwca 1993 r.

ANNA KOSMACZ-CHODOROWSKA
Instytut Gospodarki Magazynowej
Centrum Kodów Kreskowych
Poznań

Kody kreskowe i automatyczna identyfikacja dla potrzeb EDI

Stoimy na progu nowej ery – tak można by rzec mając na myśli elektroniczną wymianę danych (EDI), czyli mało wciąż jeszcze u nas znaną formę zawierania i prowadzenia kontraktów handlowych. Ta nowa forma wymiany bezpapierowych dokumentów handlowych, a dokładniej – wymiana standardowo sformatowanych danych między systemami informatycznymi partnerów handlowych przy minimalnej interwencji człowieka – jest wspierana inną nowoczesną techniką przekazywania informacji: automatyczną identyfikacją danych.

W praktyce, w zależności od zastosowania i konkretnych uwarunkowań, stosuje się określone metody automatycznej identyfikacji, do których należą:
- kody kreskowe (*Bar Code*),
- metody paska magnetycznego (*Magnetic Stripe*),
- korzystanie z fal radiowych (*Radio Frequency*),
- rozpoznawanie głosu (*Voice Recognition*),
- metoda najnowsza jaką jest rozpoznawanie obrazu (*Vision Systems*).

Automatyczna identyfikacja danych ((AI) za pomocą kodów kreskowych jest najpopularniejszą z wyżej wymienionych metod i właśnie jej zostanie poświęcony niniejszy artykuł. AI metoda kodów kreskowych jest bowiem tą techniką, która wspomaga EDI.

Oznacza to, że niektóre dane są wprowadzane do systemu EDI również przy minimalnej interwencji człowieka. Automatyczna identyfikacja zastępuje ręczne (przez klawiaturę) wprowadzanie danych – ich wczytywaniem w sposób automatyczny, przy czym kody kreskowe służą do identyfikacji tych danych za pomocą automatycznie odczytywanego i interpretowanego symbolu. Odczytywanie oraz interpretacja tego symbolu odbywa się za pośrednictwem wyspecjalizowanych urządzeń elektronicznych (najczęściej z wmontowanym dekoderem), zwanych czytnikami kodów kreskowych lub skanerami.

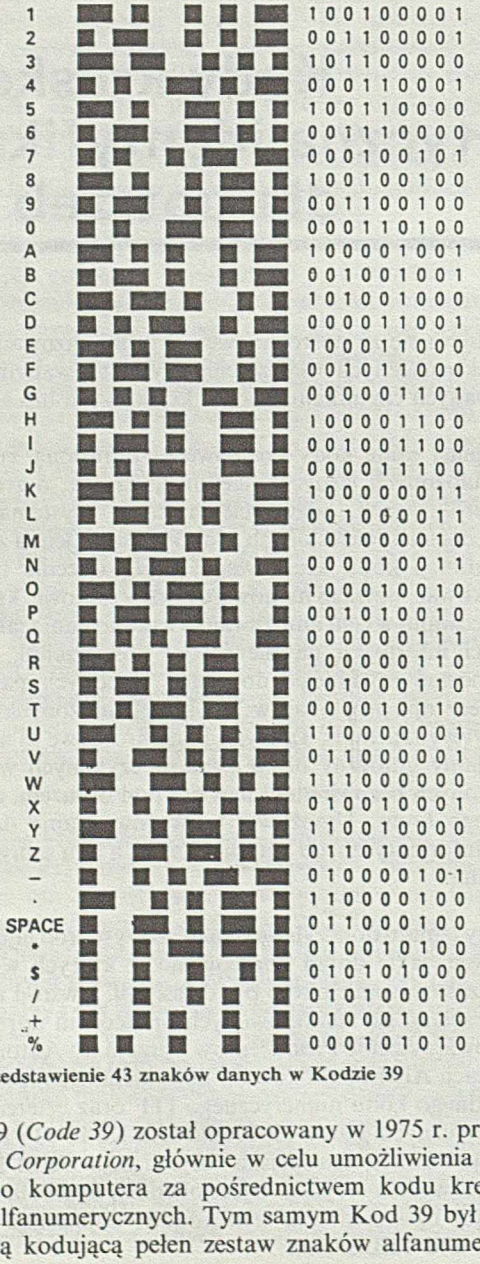
Natomiast same kody kreskowe są określoną kombinacją liniowo ułożonych jasnych i ciemnych kresek, o zróżnicowanych szerokościach, odzwierciedlającą w usystematyzowany sposób, ciąg ściśle określonych znaków. Jest to jedna z najprostszych definicji kodu kreskowego, którą obecnie należy już zmodyfikować, bowiem najnowsze kody kreskowe, które ostatnio opracowano w celu umożliwienia zakodowania jak największej ilości informacji na niewielkiej powierzchni, mają nie zawsze postać równolegle do siebie ułożonych pasków czy kresek, lecz np. prostokątów lub kresek ułożonych piętrowo (kody dwuwymiarowe). Tak więc kody kreskowe to nic innego, jak specjalny graficzny obraz znaków czytelnych wzrokowo, umieszczonych bezpośrednio nad lub pod obrazem, czyli symbolem tego kodu. Umożliwia to wprowadzanie danych do systemu informatycznego automatycznie, a tym samym szybko i bezbłędnie.

Znawcy tematu doliczyli się ponad czterysta rodzajów kodów kreskowych, jakie dotąd opracowano, z których w praktyce znalazło zastosowanie około pięćdziesięciu, a wśród nich tylko kilka doczekało się zunifikowanych opracowań normalizacyjnych. Stowarzyszenie Producentów Sprzętu do Automatycznej Identyfikacji AIM opublikowało dokumentację pięciu kodów, w tym jednego kodu numerycznego ITF oraz czterech kodów alfanumerycznych: Kodu 39, Kodabaru, Kodu 93 i Kodu 128. Na tej podstawie europejska organizacja AIM-EUROPE w 1986 r. wydała Wykaz Ujednoliconej Symboliki – USS, zawierający opis symbolik ww. kodów, dołączając do niego w 1990 r. kod TELEPEN.

Opis symbolik kodów handlowych i zasady ich stosowania określają: dla systemu UPC (*Universal Product Code*), funkcjonującego w Stanach Zjednoczonych i Kanadzie – organizacja UCC, a dla systemu EAN (*European Article Numbering*) – międzynarodowa organizacja EAN International z siedzibą w Brukseli. Jej członkiem jest właśnie Centrum Kodów Kreskowych w Poznaniu, pełniące funkcje organizacji narodowej ds. znakowania towarów kodem EAN w Polsce i reprezentujące wszystkich krajowych producentów, dystrybutorów, wydawców i inne jednostki kodujące w systemie EAN.

W gospodarce światowej funkcjonuje wiele standardów EDI, zarówno w handlu jak i w przemyśle. Od wielu lat, zwłaszcza w Ameryce, liczne branżowe organizacje standaryzujące opracowują i zarządzają systemami EDI, wykorzystującymi automatyczną identyfikację opartą na określonych symbolikach kodów kreskowych.

W zastosowaniach przemysłowych najczęściej wykorzystywanym do tych celów jest Kod 39. Również w niektórych europejskich standardach EDI, jak np. w ODETTE, Kod 39 znajduje zastosowanie. Dlatego też poniżej zamieszczam zestaw podstawowych informacji na temat tego kodu.

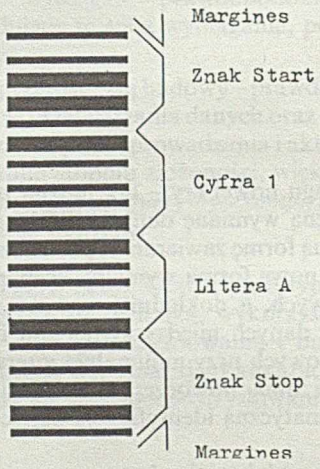


Rys. 1. Przedstawienie 43 znaków danych w Kodzie 39

Kod 39 (*Code 39*) został opracowany w 1975 r. przez firmę *Intermec Corporation*, głównie w celu umożliwienia wprowadzania do komputera za pośrednictwem kodu kreskowego danych alfanumerycznych. Tym samym Kod 39 był pierwszą symboliką kodującą pełen zestaw znaków alfanumerycznych i znalazł szerokie zastosowanie w przemyśle, zwłaszcza po 1981 r., gdy na polecenie Departamentu Obrony USA (DOD) wybrano go do oznaczania wszystkich dostaw w przemyśle wojskowym. Kod 39 zawiera w swoim zbiorze 43 znaki danych, a dzięki odpowiedniej kombinacji czterech znaków specjalnych i dużych liter, w kodzie tym można przedstawić pośrednio 128 znaków zestawu ASCII. Kod ten jest kodem dyskretnym, o dwóch szerokościach kresek. Symbol Kodu 39 rozpoczyna i kończy charakterystyczny znak gwiazdki.

Rys. 2. Przykład kodu 39

Kod ten przez ponad 10 lat był najpopularniejszym ze stosowanych kodów, zwłaszcza w przemyśle. Na jego podstawie opracowano wiele standardowych aplikacji w poszczególnych branżach (w wojsku, służbie zdrowia, przemyśle motoryzacyjnym, papierniczym, meblowym i innych) o zasięgu ogólnokrajowym, a nawet międzynarodowym.



Rys. 3. Struktura symboli Kodu 39

W przemyśle motoryzacyjnym Kod 39 stosują prawie wszyscy kooperanci, dostawcy i filie takich zakładów, jak np. Alfa Romeo, Lancia, BMW, Bosch, Citroen, Ferrari, Fiat, Ford, Iveco, Jaguar, Ferguson, Mercedes, Michelin, Nissan, Peugeot, Philips, Pirelli, Renaut, Saab Scania, Seat, Volkswagen, Volvo, Zanussi – wymieniając chociażby te najbardziej znane.

Przemysłowe standardy dotyczące komunikatów EDI, określają jednocześnie sposoby znakowania towarów, w tym przede wszystkim rodzaj informacji kodowanych i bezpośrednich, a więc ich treść i format danych, a także to, które z nich powinny być przedstawiane w kodzie kreskowym (po uprzednim określeniu symboliki kodu) oraz w jakiej formie te informacje mają być prezentowane.

W jednym z najpopularniejszych przemysłowych standardów EDI – ODETTE towary są oznaczane tzw. etykietami transportowymi, gdzie część danych jest prezentowana właśnie w kodzie kreskowym 39. W standardzie tym ustalono kilka rodzajów etykiet, w zależności od wymiarów opakowania i zawartości jednostki wysyłkowej.

Standard ODETTE stosuje standardowane etykiety transportowe na małe lub wydłużone opakowania o określonej zawartości oraz, najczęściej stosowane, etykiety na opakowania typu: karton, skrzynia, paleta, kontener. W zależności od zawartości jednostek wysyłkowych stosuje się cztery podstawowe rodzaje etykiet:

- Etykietę Identyfikacji Części (*S*);
- Etykietę Główną (*S*), jeśli jednostka wysyłkowa zawiera takie same towary, uprzednio nie oznaczone etykietą;
- Etykietę Główną (*M*), gdy jednostka wysyłkowa zawiera jednakowe towary, etykietowane;
- Etykietę Główną (*G*), gdy jednostka wysyłkowa zawiera różne towary, odrębnie etykietowane.

Każda z powyższych etykiet ma ustalony format, pola informacyjne o określonej zawartości, opisane identyfikatorem pola. W pola te są wpisywane dane o formacie uzgodnionym z partnerem handlowym. Część z nich jest danymi bezpośrednimi, np. daty, waga netto, brutto, nr seryjny, a część z nich jest danymi zakodowanymi. Zarówno dane bezpośrednie, jak i kodowane, są przedstawione w sposób czytelny wzrokowo oraz w postaci symbolu kodu kreskowego. Określona grupa danych powinna być zawsze przedstawiona graficznie w kodzie kreskowym.

Oprócz przemysłowych standardów EDI funkcjonują również tzw. standardy handlowe. Jeden z nich ma szansę stać się dominującym w tej dziedzinie, jest bowiem pochodnym międzynarodowego standardu UN/EDIFACT. Jest to standard opracowany przez organizację EAN International dla handlu i nazwany standardem EANCOM.

Podstawowym elementem EANCOM jest międzynarodowy system znakowania towarów, w którym każdy rodzaj towaru, zarówno w postaci jednostki konsumenckiej, jak i w postaci jednostki wysyłkowej, ma swój indywidualny numer. W systemie tym również inne dane dotyczące towaru są oznaczane w sposób standardowy, np. dane dotyczące partnerów handlowych (identyfikatory dostawców i odbiorców – tzw. kody lokalizacyjne) oraz cały zestaw danych uzupełniających, służących do przekazywania dodatkowych informacji o towarze i innych danych niezbędnych przy zawieraniu transakcji handlowych i prowadzeniu wymiany towarowej.

W systemie EAN, będącym pochodną amerykańskiego systemu UPC, towary w postaci jednostek konsumenckich (w handlu detalicznym) są znakowane kodem numerycznym EAN-13, lub – dla towarów o małych wymiarach – kodem EAN-8, które mają ściśle określoną strukturę liczby kodowanej (tabele 1 i 2).

Tabela 1. Struktura liczby kodowanej kodu EAN-13

MIEJSCA CYFR	13	12	11	10	9	8	7	6	5	4	3	2	1
STRUKTURA LICZBY KODOWANEJ	Nr kraju			Nr jednostki				Nr indywidualny towaru					cyfra kontrolna
EAN - 13	5	9	0	J_1	J_2	J_3	J_4	T_1	T_2	T_3	T_4	T_5	K

Tabela 2. Struktura liczby kodowanej kodu EAN-8

MIEJSCA CYFR	8	7	6	5	4	3	2	1
STRUKTURA LICZBY KODOWANEJ	Nr kraju			Nr indywidualny towaru				cyfra kontrolna
EAN - 8	5	9	0	T_1	T_2	T_3	T_4	K

Rys. 4. Wzór standardowej etykiety transportowej ODETTE

W Stanach Zjednoczonych i Kanadzie towary oznaczane są kodem numerycznym UPC, przy czym wyróżnia się następujące wersje:

- pełna – UPCA (12-znakowa),
- skrócona – UPC-E (6-znakowa).

Towary w jednostkach wysyłkowych, to znaczy sformatowane w celu usprawnienia procesów transportowo-magazynowych, są znakowane najczęściej kodem numerycznym ITF (tzw. przeplatanym 2 z 5). W kodzie tym struktura liczby kodowanej jest pochodną liczby kodowanej kodów EAN-13 lub EAN-8, uzupełnioną na początku jednocyfrowym identyfikatorem jednostki wysyłkowej. Identyfikator ten oznacza rodzaj i/lub poziom pakowania. Liczba kodowa ma w tym zastosowaniu 14 cyfr, stąd nazwa kodu ITF-14.

Tabela 3. Lista Standardowych Identyfikatorów Zastosowania

Lista Standardowych Identyfikatorów Zastosowania

IZ		Treść		Format
00		Seryjny numer jednostki wysyłkowej		n2+n18
01		Numer towaru EAN/Numer jednostki wysyłkowej		n2+n14
02		Numer EAN towarów zawartych w innej jednostce		n2+n14
10		Numer serii produkcyjnej		n2+an..20
11		Data pakowania		n2+n6
13		Data pakowania		n2+n6
15		Minimalny okres trwałości (RRMMDD)		n2+n6
17		Maksymalny okres trwałości (RRMMDD)		n2+n6
20		Wariant produktu		n2+n2
21		Seryjny numer towaru		n2+an..20
22		HIBCC - oznakowanie towaru dla służby zdrowia		n2+an..29
23	(*)	Numer partii (przejściowo)		n3+n..19
30		Zmienna ilość		n2+n..8
310	(**)	Waga netto [kg]		n4+n6
311	(**)	Długość lub pierwszy wymiar [m]		n4+n6
312	(**)	Szerokość, średnica lub drugi wymiar [m]		n4+n6
313	(**)	Głębokość, grubość wysokość lub trzeci wymiar [m]	dla potrzeb handlu	n4+n6
314	(**)	Powierzchnia [m^2]		n4+n6
315	(**)	Objętość netto [dm^3]		n4+n6
316	(**)	Objętość netto [m^3]		n4+n6
320	(**)	Waga netto [funty]		n4+n6
330	(**)	Waga brutto [kg]		n4+n6
331	(**)	Długość lub pierwszy wymiar [m]		n4+n6
332	(**)	Szerokość, średnica lub drugi wymiar [m]		n4+n6
333	(**)	Głębokość, grubość, wysokość lub trzeci wymiar [m]	dla potrzeb transportu	n4+n6
334	(**)	Powierzchnia [m^3]		n4+n6
335	(**)	Objętość [dm^3]		n4+n6
336	(**)	Objętość [m^3]		n4+n6
340	(**)	Waga netto [funty]		n4+n6
37		Ilość		n2+n..8
400		Numer zlecenia zakupu		n3+an..30
410		Kod lokalizacyjny odbiorcy towaru		n3+n13
411		Kod lokalizacyjny odbiorcy faktury / rachunku		n3+n13
412		Kod lokalizacyjny dostawcy		n3+n13
420		Kod pocztowy odbiorcy krajowego		n3+an..9
421		Kod pocztowy odbiorcy zagranicznego		n3+n3+an..9
8001		Oznaczenie towaru w rolkach		n4+n14
8002		Numer seryjny dla telefonów komórkowych		n4+an..20
90		Inne zastosowania wewnętrzne		n2+an..30
91, 92		Zastosowania wewnętrzne - dla dostawców surowców, dodatków i materiałów na opakowania lub opakowań		n2+an..30
93, 94		Zastosowania wewnętrzne - dla producentów towarów		n2+an..30
95, 96		Zastosowania wewnętrzne - dla spedytorów		n2+an..30
97, 98		Zastosowania wewnętrzne - dla hurtowników i detalistów		n2+an..30
99		Dowolny tekst		n2+an..30

(*) : Plus jedna cyfra oznaczająca liczbę znaków
(**) : Plus jedna cyfra oznaczająca miejsce dziesiętne

Jak już wspomniano, oprócz jednoznacznej w skali światowej identyfikacji towarów w systemie EAN, wiele innych danych jest kodowanych w innym określonym formacie, aktualny pełen ich wykaz zawiera tabela 3. System znakowania tych danych został przyjęty wspólnie przez organizacje EAN International oraz amerykańską UCC.

Rys. 5. Wzór Kodu 128

W celu przedstawienia tych danych w postaci umożliwiającej ich automatyczny odczyt, organizacje te wybrały alfanumeryczny Kod 128. Kod ten jest kodem ciągłym (o budowie modularnej), umożliwiającym bezpośrednie zakodowanie 128 znaków zestawu ASCII, czym w znacznym stopniu przewyższa Kod 39. Główną jego zaletą jest to, że tę samą liczbę znaków można zakodować w krótszym symbolu.

Rys. 6. Graficzne odwzorowanie tego samego ciągu znaków w Kodzie 39 i Kodzie 128 (o 44% krótszym)

W porozumieniu ze stowarzyszeniem AIM w Kodzie-128 zarezerwowano do wyłącznej dyspozycji systemu EAN jeden ze znaków funkcyjnych – FNC 1. Znak ten umieszczony bezpośrednio po znaku start tego kodu, wyróżnia znakowanie danych w systemie EAN od jakiegokolwiek innego zastosowania. Tym samym powstał jak gdyby, odrębny kod – UCC/EAN-128, zwany krótko EAN-128.

Obecnie organizacja EAN International wraz z innymi, branżowymi organizacjami standaryzacyjnymi w przemyśle samochodowym, metalowym i elektronicznym prowadzą prace nad opracowaniem zunifikowanej etykiety międzybranżowej, w której dane byłyby odzwierciedlane w kodzie EAN-128. Są to jednak procesy długotrwałe, zarówno ze względu na ogrom pracy, jak i konieczność przekonania wielu osób do celowości zmian.

Podsumowując należy stwierdzić, że standardy znakowania (kodowania) towarów są integralną częścią wymiany danych, a stosowanie automatycznej identyfikacji przez kody kreskowe znacznie zwiększa podstawowe zalety EDI.

MEMBER OF EAN INTERNATIONAL

Znak Międzynarodowego Stowarzyszenia Kodów Kreskowych

INFORMATYKI nie ma w sprzedaży kioskowej, istnieje tylko w prenumeracie. Można ją zaprenumerować na wszystkich pocztach w kraju.

Jednak każdy numer czasopisma można kupić w Redakcji. Prowadzimy sprzedaż odręczną i wysyłkową. Wystarczy do nas zadzwonić (Warszawa tel. 39-14-34).

Dla spóźnionych rezerwujemy do sprzedaży egzemplarze począwszy od numeru styczniowego br.

Na życzenie odbiorcy wystawiamy fakturę z podatkiem VAT.

I Krajowa Konferencja Elektronicznej Wymiany Danych
Arturówek k. Łodzi, 15–17 czerwca 1993 r.

ELŻBIETA HAŁAS
Instytut Gospodarki Magazynowej
Centrum Kodów Kreskowych
Poznań

EANCOM – praktyczny przewodnik wdrażania zasad UN/EDIFACT w handlu i przemyśle

Możliwość efektywnego i szybkiego porozumiewania się, to kluczowa sprawa w świecie biznesu. Międzynarodowe stowarzyszenie EAN INTERNATIONAL od 1977 r. wdraża i rozwija jednolity system identyfikowania towarów i usług, który gwarantuje jednoznaczność przekazywanych danych. System ten, znany jako system EAN, jest bardzo popularny na całym świecie, a także w Polsce, gdzie na wielu towarach można już zauważyć charakterystyczny wzór kodu kreskowego. Organizacja EAN INTERNATIONAL liczy obecnie 54 członków, a Polska jest w niej reprezentowana przez Centrum Kodów Kreskowych w Poznaniu.

Kod kreskowy i automatyczna identyfikacja towarów to nie jedyne pola działalności Stowarzyszenia EAN INTERNATIONAL. Jego członkowie bardzo szybko zdali sobie sprawę z tego, jak silnym narzędziem może być kod kreskowy w elektronicznej wymianie danych i jak bardzo zwiększa on jej efektywność. W 1987 r. Stowarzyszenie opracowało standard EDI – EANCOM (**EAN–COM**munication), który wykorzystuje zasady składni normy UN/EDIFACT i system identyfikowania towarów EAN do projektowania elektronicznych komunikatów, odpowiedników dokumentów handlowych dla potrzeb handlu i przemysłu.

EANCOM i EDIFACT

EANCOM to praktyczny i szczegółowy przewodnik wdrażania zasad normy UN/EDIFACT w handlu i przemyśle. Komunikaty UN/EDIFACT, będąc efektem kompromisu między użytkownikami różnych standardów narodowych i branżowych, są bardzo skomplikowane i często zbyt złożone jak na potrzeby przeciętnego przedsiębiorstwa. Z doświadczenia wielu firm, które już wykorzystują EDI przy realizacji transakcji handlowych wynika, że do przesyłania danych wystarczyłaby znacznie prostsza postać elektronicznej faktury, zamówienia lub innego dokumentu. Praktycznie wykorzystują one zaledwie 25% informacji zamieszczanych w komunikatach UN/EDIFACT. EANCOM nie tylko wybiera te segmenty i elementy danych, które mają zastosowanie w handlu, ale także dostarcza jasnych definicji, wyjaśnień i praktycznych przykładów, dzięki czemu elektroniczna wymiana dokumentów handlowych staje się prosta, jasna i efektywna.

EANCOM opiera się na standardowych komunikatach UN/EDIFACT i proponuje szczegółowe projekty poszczególnych wiadomości. EANCOM ogranicza się wyłącznie do ustalenia struktury komunikatów wykorzystywanych w handlu i w przemyśle, ale dokładnie informuje, które elementy danych i kody należy zastosować z bogatej biblioteki zalecanej przez UN/EDIFACT. Stowarzyszenie zajmuje się także opracowaniem nowych komunikatów, jeszcze niezdefiniowanych w ramach normy UN/EDIFACT, a niezbędnych w handlu detalicznym i hurtowym. W ten sposób postępują prace nad komunikatami *Raport Sprzedaży* oraz *Prognoza Zbytu*.

Główne cechy EANCOM

EANCOM nie został opracowany dla jednej konkretnej branży czy gałęzi przemysłu. Jest to praktyczny, wielobranżowy poradnik, który może być stosowany zarówno w przypadku transakcji krajowych jak i międzynarodowych. EANCOM wykorzystuje w swoich komunikatach następujące standardy EAN:

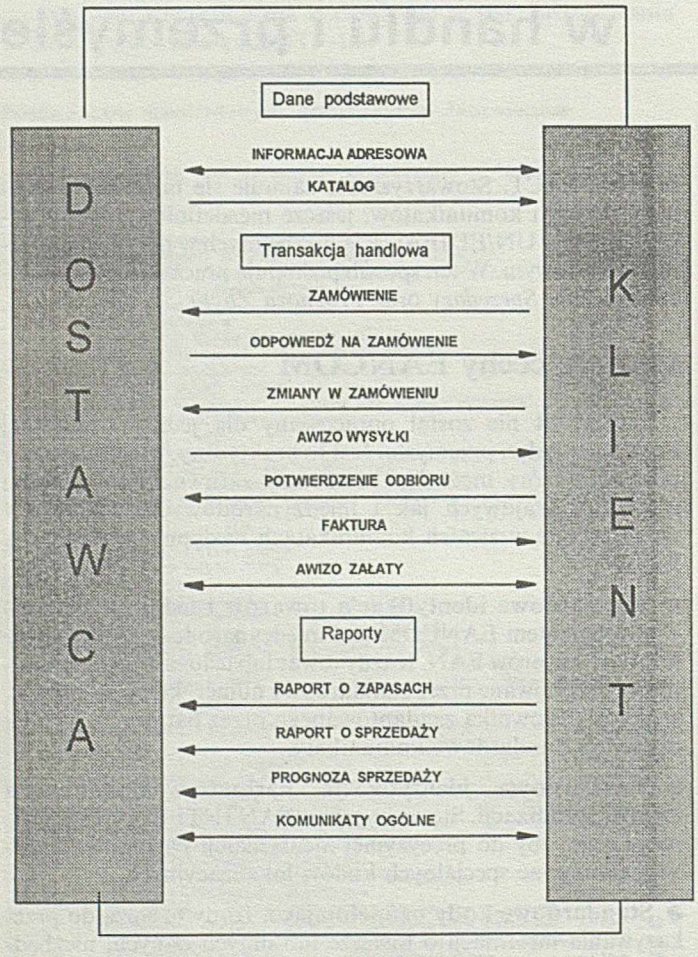
- **Standardowa identyfikacja towarów i usług.** Podstawowym elementem EANCOM jest międzynarodowy system znakowania towarów EAN. Każdy towar lub usługa są jednoznacznie identyfikowane przez standardowy numer, który jest częścią wspólnego słownika zaadaptowanego przez partnerów wymieniających standardowe komunikaty.
- **Standardowa identyfikacja partnerów handlowych i miejsc lokalizacji.** Stowarzyszenie EAN INTERNATIONAL proponuje, aby do precyzyjnej identyfikacji partnerów handlowych używać specjalnych kodów lokalizacyjnych.
- **Standardowe kody uzupełniające.** Kody te służą do przekazywania informacji o towarze lub innych danych, niezbędnych przy zawieraniu transakcji.
- **Standardowe formaty kodów kreskowych.** Zastosowania standardowych formatów kodów kreskowych, rekomendowanych przez EAN INTERNATIONAL umożliwia automatyczną i bezpieczną rejestrację towarów oraz dodatkowych danych.
- **Standardowy format dla transakcji handlowych.** Format ten dotyczy tych transakcji, które są zawierane za pomocą EDI, a więc bezpośrednio między systemami informatycznymi partnerów handlowych.

EDI uwydatnia konieczność pełnej identyfikacji kodami EAN wszystkich asortymentów towarów wraz z ich formami specjalnymi (np. wariantami promocyjnymi) i rodzajami opakowań. Opierając się na zasadach EAN, EANCOM proponuje spójne reguły kodowania w celu jednoznacznej identyfikacji dla:

- opakowań spedycyjnych,
- produktów o zmiennej ilości,
- grup towarów,
- wariantów promocyjnych towarów,
- opakowań zwrotnych.

Przepływ informacji w EANCOM

EANCOM został opracowany głównie z myślą o tych firmach, które zawierają dziennie wiele transakcji ze swoimi, zazwyczaj stałymi, partnerami handlowymi. Z tego względu EANCOM zamiast pojedynczych komunikatów proponuje wymianę sekwencji komunikatów w celu uniknięcia powtarzania ciągle tych samych danych. Dzięki temu komunikaty EANCOM są bardzo krótkie i prawie całkowicie pozbawione informacji w postaci tekstowej, co z kolei umożliwia ich w pełni automatyczne przetwarzanie. Dostępne w EANCOM komunikaty (rys. 1) można podzielić na następujące kategorie:



Rys. 1. Przepływ informacji w EANCOM

- **Dane podstawowe:** informacja adresowa, katalog. Komunikaty te zawierają dane wymieniane między partnerami handlowymi i przechowywane w pamięci komputera jako dane referencyjne przy kolejnych transakcjach. Informacja adresowa służy do identyfikowania wszystkich występujących w transakcji miejsc lokalizacji. Katalog dostarcza klientowi wszelkich danych o oferowanym towarze.
- **Transakcje handlowe.** W tej kategorii mieszczą się wszystkie komunikaty obsługujące pełny cykl transakcji (zamówienie–sprzedaż–zapłata–dostawa). Zamówienie obejmuje wszystkie komunikaty związane z procesem składania i potwierdzenia zamówienia. Zawiadomienie o wysyłce oraz zawiadomienie o przyjęciu są związane z fizyczną wysyłką i dostawą towarów. Zawiadomienie o wysyłce pozwala zawiadomić kupującego z wyprzedzeniem o dostawie, podając wszystkie szczegóły dotyczące dostarczanego towaru. Zawiadomienie o przyjęciu potwierdza przyjęcie towaru lub informuje o zastrzeżeniach. Faktura oraz zawiadomienie o zapłacie dotyczą realizacji płatności za dostarczony towar.
- **Raporty.** Komunikaty tej kategorii umożliwiają wymianę informacji o planach firmy, co znacznie ułatwia zrozumienie wzajemnych oczekiwań. Raport o sprzedaży dostarcza istotnych informacji marketingowych dotyczących tempa i wielkości sprzedaży towarów dostawcy. Prognoza sprzedaży pozwala dostawcy oszacować przyszły popyt i odpowiednio zaprogramować produkcję. Raport o zapasach dostarcza aktualnych danych o wielkości zapasów, co pozwala na lepszą kontrolę dystrybucji oraz przewidzenie wymagań dotyczących dostaw.

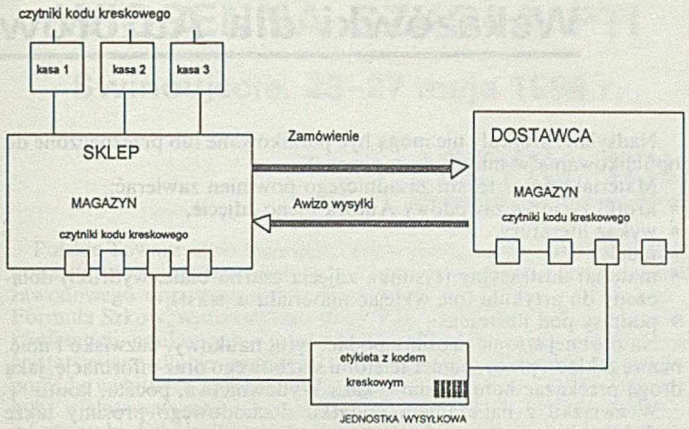
EANCOM na świecie

EANCOM jako podstandard EDIFACT staje się coraz popularniejszy nie tylko w Europie, ale i na całym świecie. Wiele państw, mimo posiadania własnych standardów, przystąpiło do popularyzacji EANCOM, zwłaszcza w środowisku właścicieli sklepów i hurtowni. Niektóre z nich, np. Dania przyjęły EANCOM jako jedyny krajowy standard EDI. Obecnie wszystkie firmy komputerowe opracowujące konwertery EDI uwzględniają w swoich oprogramowaniach EANCOM. Międzynarodową pozycję EANCOM wśród innych standardów EDI ilustruje tabela. Tabela ta została opracowana przez EAN INTERNATIONAL na podstawie danych zebranych przez organizacje narodowe i prezentuje istniejącą w 1992 r. i przewidywaną na 1993 r. liczbę użytkowników EDI.

Miejsce EANCOM wśród innych standardów EDI

KRAJ	Nazwa krajowego standardu	1992	1993	Nazwa projektu EANCOM	1992	1993
Australia	ANSI X.12 EDIFACT	2000-3000	3000-4000	-	-	-
Austria	SEDAS	400	450	EANCOM Ecodex	15	40
Belgia/ Luksemburg	ICOM	101	135	EANCOM	27	50
Dania	-	-	-	HANCOM	180	300
Finlandia	EDIFACT	100	400	-	-	-
Francja	GENCOD Allegro	1500	1750	ALLEGRO	6	12
Hiszpania	AECOM	200	350	EAN Concorde (TEDIS)	3	4
Holandia	TRANSCOM	700	900	EANCOM	150	500
Irlandia	-	-	-	EANCOM	125	250
Islandia	-	-	-	EANCOM ISEDI	6	30
Kolumbia	-	-	-	EANCOM	5	25
Niemcy	SEDAS, SINFOS, MADAKOM	700	780	Przejście SEDAS/EANCOM (TEDIS)	8	20
Norwegia	STANDARD RECORDS	2050	2200	EANCOM	30	50
Nowa Zelandia	-	-	-	EANCOM	60	-
Szwajcaria	-	-	-	nEANder	54	80
Szwecja	-	-	-	EDIT	50	200
Wielka Brytania	TRADACOMS	6000	10000	UK EDIFACT Trade Message	200	1000
Włochy	-	-	-	EANCOM	50	150
RAZEM		**14251**	**20515**		**964**	**2788**

Inicjatywę zaadaptowania EDI do istniejących systemów wykorzystujących kody kreskowe podjęły firmy działające w obszarze produkcji i handlu towarami przeznaczonymi do sprzedaży detalicznej oraz w przemyśle samochodowym. Producenci sprzętu do automatycznej identyfikacji uważają, że nie można wykorzystać wszystkich zalet EDI bez zastosowania kodów kreskowych. Przykład wykorzystania obu technologii



Rys. 2. Wykorzystanie kodu kreskowego i EDI

ilustruje rys. 2. Na podstawie danych o sprzedaży (zebranych przez kasy) oraz stanu magazynu, komputer sporządza zamówienie, które zostaje następnie przesłane do stałego dostawcy. W momencie wysyłki towarów, dostawca używając tej samej sieci EDI przesyła awizo wysyłki, które dokładnie opisuje przesyłane towary. Kody kreskowe na każdej jednostce wysyłkowej identyfikują jej zawartość dla pracowników działu zaopatrzenia w sklepie. Po odczytaniu etykiety towary są kierowane bezpośrednio do sklepu lub do magazynu.

Do największych firm uczestniczących w pracach nad EDIFACT i EANCOM należą m.in.: BALLY, BARILA, CARLSBERG BREWERIES, COCA-COLA, COLGATE-PALMOLIVE, KNORR, LEVI STRAUSS & CO, METRO, NESTLE, OSRAM, PHILIPS, PROCTER & GAMBLE.

LITERATURA

Materiały źródłowe EAN INTERNATIONAL.

Ogłoszenia prosimy zgłaszać:

pisemnie
Red. INFORMATYKI
Pl. Inwalidów 10, p. 104, 01-552 Warszawa
lub
Dział Reklamy i Marketingu
ul. Mazowiecka 12, 00-950 Warszawa
skr. poczt. 1004

telefonicznie
nr 39-14-34 lub 27-43-66
telefaksem
nr 26-80-16
teleksem
814877
z zaznaczeniem – Red. INFORMATYKI

PIERWSZY POLSKI INTELIGENTNY
SYSTEM INFORMATYCZNY WSPOMAGAJĄCY ZARZĄDZANIE
PRODUKCJĄ - (TPPiRP), FINANSAMI - KOSZTAMI, SPRZEDAŻĄ, KADRAMI
WRAZ Z SYSTEMEM INFORMACJI KIEROWNICTWA
WSPOMAGANY SYSTEMEM SZTUCZNEJ INTELIGENCJI PC-SHELL
ORAZ EKSPERCKĄ BAZĄ WIEDZY EKONOMICZNO-FINANSOWEJ - ISAF
(DLA DUŻYCH PRZEDSIĘBIORSTW PRODUKCYJNYCH)

Perfect-Ekspert AITECH

(DZIAŁAJĄCY W OPARCIU O ŚRODOWISKO PRACY - WYŁĄCZNIE SYSTEM OPERACYJNY DOS, NetWare v3.11, v4.01 FIRMY NOVELL)

OFERUJE

PRZEDSIĘBIORSTWO PRODUKCYJNO-USŁUGOWE INNOWACJI TECHNICZNYCH

JUNISOFTEX Sp. z o.o.

ROK ZAŁOŻENIA 1987
WARTOŚĆ SPRZEDAŻY wg F-02 1992 (14,6 mld), wg F-01 na 1993.10.31 (15,4 mld)
TEL/FAX 315-131, 315-700, 321-525, 388-052, 388-054 TLX 036233
BIURO TECHNICZNO-HANDLOWE
44-100 GLIWICE ul. Kościuszki 1
WSZYSTKICH ZAINTERESOWANYCH ZAPRASZAMY NA MERYTORYCZNE ROZMOWY DO NASZEGO BIURA TECHNICZNO-HANDLOWEGO
OFERUJEMY RÓWNIEŻ INTELIGENTNE NARZĘDZIE DO PRODUKCJI EKSPERCKICH BAZ WIEDZY

PC-SHELL

ARTIFICIAL INTELLIGENCE

DLA JEDNOSTEK BUDŻETOWYCH (NAUKA, OŚWIATA, URZĘDY MIAST I GMIN, SŁUŻBA ZDROWIA) BEZ PRAWA DO KOMERCYJNEGO WYKORZYSTYWANIA W CENIE 5.000.000,- (BEZ VAT) - INFORMACJE I ZAMÓWIENIA PROSIMY SKŁADAĆ BEZPOŚREDNIO U PRODUCENTA ARTIFICIAL INTELLIGENCE LABORATORY - AITECH, KATOWICE 14 SKR. POCZT. 2804.
DLA POZOSTAŁYCH JEDNOSTEK GOSPODARCZYCH BEZ PRAWA KOMERCYJNEGO WYKORZYSTYWANIA TJ. WYŁĄCZNIE NA WŁASNE POTRZEBY DO KOŃCA I KW. 1994 ROKU W CENIE PROMOCYJNEJ 10.000.000,- (BEZ VAT), DLA KUPUJĄCYCH DO CELÓW KOMERCYJNYCH DO KOŃCA I KW. 1994 25.000.000,- (BEZ VAT), LICENCJA OGRANICZONA NA NIEOKREŚLONĄ ILOŚĆ ODBIORCÓW KOŃCOWYCH.
POSZUKUJEMY DO WSPÓŁPRACY LUB NAWIĄŻEMY WSPÓŁPRACĘ Z EKSPERTAMI Z DOWOLNYCH DZIEDZIN NAUKI I TECHNIKI.

0/3/93

Dedukcyjne bazy danych

dokończenie ze s. 7

LITERATURA

[1] Bayer P., Lefebvre A., Vieille L.: Architecture and Design of the EKS Deductive Database System. VLDB Journal on Prototypes of Deductive Database Systems, 1993

[2] Das S. K.: Deductive Databases and Logic Programming. Adison-Wesley Publishing Company, 1992

[3] Date C. J.: An Introduction to Database System, Volume II. Adison-Wesley Publishing Company, 1981

[4] Phipps G.: Glue-Nail: A Deductive Database System. ACM-SIGMOD Conf. on Management of Data, Denver, Colorado, 1991

[5] Ramakrishnan R.: Bothner P., Srivastava D., Sudarshan S.: CORAL – A Database Programming Language, NACPL 90 Workshop on Deductive Databases, Kansas State University, 1990

[6] Srivastava D., Ramakrishnan R., Seshadri P., Sudarshan S.: Coral$_{++}$: Adding Object-Orientation to a Logic Database Language. VLDB 93 Conference Proc., Dublin, 1993

[7] Tsur S., Naqvi S.: A Logical Language for Data and Knowledge Bases. Computer Science Press, New York, 1989

[8] Vaghani J., Ramamohanarao K., Kemp S., Somogyi Z., Stuickey P.: Design Overview of the Aditi Deductive Database System. NACLP 90 Workshop on Deductive Databases, Kansas State University, 1990

[9] Wieczerzycki W.: Od relacyjnych do obiektowych baz danych. Informatyka, styczeń 1993

[10] Vieille L., Bayer P., Kuchenchoff V., Lefebvre A.: EKS-V1, A Short Overview. AAAI Workshop on Knowledge Base Management System, Boston, 1990.

Wskazówki dla Autorów

Nadsyłane artykuły nie mogą być publikowane lub przeznaczone do opublikowania w innych czasopismach.

Materiał oprócz tekstu zasadniczego powinien zawierać:

- ★ krótki życiorys zawodowy Autora i jego zdjęcie,
- ★ wykaz literatury,
- ★ tabele,
- ★ materiał ilustracyjny (rysunki, zdjęcia czarno-białe, wydruki) dołączony do artykułu (nie wklejać materiału w tekst),
- ★ podpisy pod ilustracje.

Na osobnej stronie prosimy podać: tytuł naukowy, nazwisko i imię, nazwę zakładu pracy, numer telefonu służbowego oraz informację, jaką drogą przekazać honorarium – kasa Wydawnictwa, poczta, konto.

W związku z naliczaniem podatku dochodowego prosimy także podać:

- ★ oprócz nazwiska i imienia – także drugie imię,
- ★ imiona rodziców,
- ★ datę i miejce urodzenia,
- ★ numer identyfikacyjny PESEL,
- ★ miejsce zameldowania,
- ★ adres Urzędu Skarbowego właściwego dla miejsca zamieszkania Autora.

Tekst artykułu prosimy dostarczyć w formie maszynopisu lub wydruku komputerowego, pisanego jednostronnie, z podwójnym odstępem (30 wierszy na stronie i 60 znaków w wierszu).

Rysunki powinny być czytelne i mogą być wykonane ołówkiem.

Autor opublikowanego artykułu otrzymuje honorarium i bezpłatny egzemplarz okazowy.

Materiałów nie zamówionych redakcja nie zwraca.

Wieczerzycki W.: Dedukcyjne bazy danych INFORMATYKA 1994, nr 2, s. 2 Charakterystyka budowy oraz sposobu funkcjonowania i wykorzystywania dedukcyjnych baz danych, a także ich porównanie z systemami programowania w logice oraz z relacyjnymi bazami danych.	**Wieczerzycki W.: Deductive databases** INFORMATYKA 1994, No. 2, p. 2 Structure, operating mode and utilization of deductive databases, as well as their comparison with programming in logic and with relative databases.	**Wieczerzycki W.: Deduktionsdatenbanken** INFORMATYKA 1994, Nr. 2, S. 2 Eine Charakteristik von Konstruktion, Betrieb und Benutzung der Deduktionsdatenbanken, sowie ein Vergleich mit Programmierungssystemen in der Logik und mit Relationsdatenbanken.
Simiński R.: Projekt i realizacja obiektowo zorientowanego podsystemu komunikacji z użytkownikiem INFORMATYKA 1994, nr 2, s. 8 Zadania oraz koncepcja rozwiązań podsystemu komunikacji z użytkownikiem z wykorzystaniem techniki projektowania obiektowego.	**Simiński R.: Design and realization of object oriented subsystem for communication with user** INFORMATYKA 1994, No. 2, p. 8 Tasks and solution idea of an user communication subsystem utilising object programming techniques.	**Simiński R.: Projekt und Realisierung eines objektorientierten Untersystems für Kommunikation mit Benutzer** INFORMATYKA 1994, Nr. 2, S. 8 Aufgaben und Lösungskonzeption von einem Untersystem für Kommunikation mit Benutzer mit Berücksichtigung der Objektprojektierungstechnik.
Hołodnik-Janczura G., Płonka-Szydlak E., Zabawska E.: Realizacja przedsięwzięć informatycznych z wykorzystaniem metodyki CASE firmy Oracle INFORMATYKA 1994, nr 2, s. 15 Charakterystyka zespołu narzędzi do realizacji pełnego cyklu życia systemu informatycznego z zastosowaniem opracowanej przez firmę Oracle metodyki CASE*Method.	**Hołodnik-Janczura G., Płonka-Szydlak E., Zabawska E.: Realization of data processing undertakings utilizing the Oracle's CASE methodology** INFORMATYKA 1994, No. 2, p. 15 Characteristics of the tool set for realization of the complete data processing system life cycle, which was elaborated by Oracle applying the CASE*Method.	**Hołodnik-Janczura G., Płonka-Szydlak E., Zabawska E.: Realisierung von Informatikunternehmungen mit Ausnutzung der CA-SE-Methodik von Firma Oracle** INFORMATYKA 1994, Nr. 2, S. 15 Eine Charakteristik von Hilfsmittelgruppe für Realisierung eines vollen EDV-Systemlebenszyklus mit Verwendung der von der Firma Oracle erarbeiteten CASE-Methode.
Kosmacz-Chodorowska A.: Kody kreskowe i automatyczna identyfikacja dla potrzeb EDI INFORMATYKA 1994, nr 2, s. 21 Omówienie najważniejszych rodzajów kodów kreskowych stosowanych w elektronicznej wymianie danych (EDI) do automatycznej identyfikacji towarów i innych danych.	**Kosmacz-Chodorowska A.: Bar codes and automatic identification for EDI needs** INFORMATYKA 1994, No. 2, p. 21 Discussion of main bar code kinds, which are applied in Electronic Data Interchange (EDI) for automatic identification of merchandise and other data.	**Kosmacz-Chodorowska A.: Streifencodes und automatisierte Identifikation für EDI-Bedürfnisse** INFORMATYKA 1994, Nr. 2, S. 21 Eine Besprechung von wichtigsten Streifecodevarianten, die im elektronischen Datenaustausch (EDI) für Identifizierung von Waren oder sonstigen Daten angewendet werden.
Hałas E.: EANCOM – praktyczny przewodnik wdrażania zasad UN/EDIFACT w handlu i przemyśle INFORMATYKA 1994, nr 2, s. 25 Charakterystyka standardu EANCOM, opartego na międzynarodowym systemie identyfikacji towarów EAN i pełniącego funkcję poradnika przy projektowaniu komunikatów EDI.	**Hałas E.: EANCOM – a practical guide for the UN/EDIFACT principles implementation in commerce and industry** INFORMATYKA 1994, No. 2, p. 25 Characteristics of the EANCOM standard, based on EAN international system for merchandise identification, which is a practical guide for EDI message design.	**Hałas E.: EANCOM – ein praktischer Wegweiser für Einsatz der UN/EDIFACT-Grundsätze in Handel und Industrie** INFORMATYKA 1994, Nr. 2, S. 25 Eine Charakteristik von dem auf EAN-Warenidentifizierungssystem basierten EANCOM-Standard, der die Funktion eines Wegweisers bei der Projektierung von EDI-Mitteilungen, erfüllt.

WIOSENNA SZKOŁA PTI

Świnoujście, 23–27 maja 1994 r.

JĘZYKI 4GL

Polskie Towarzystwo Informatyczne w ramach prowadzonej działalności statutowej wypracowało nową formułę szkolenia i doskonalenia zawodowego oraz promocji nowoczesnych produktów i technologii. Formuła Szkoły, wprowadzona przez PTI, przewiduje połączenie zalet konferencji naukowych i kursów zawodowych dając uczestnikom wystarczająco długie wykłady, w trakcie których przedstawiane są poglądy wybitnych specjalistów na poruszany temat. Szkoły Jesienne były wielotematyczne i koncentrowały się wokół trendów rozwoju informatyki. Szkoły Wiosenne były monotematyczne i koncentrowały się wokół problemów metodologii tworzenia systemów informatycznych, przedstawiając poglądy różnych środowisk.

Pięć kolejnych spotkań Wiosennej Szkoły PTI w latach 1988–1992 spełniło zakładaną przez organizatorów rolę integracji środowiska informatyków wokół problemów metodyki tworzenia systemów informatycznych. Aktywny udział ponad setki uczestników w każdej z kolejnych Szkół, szeroki odzew środowiska wyrażany w bardzo różnorodny i sympatyczny sposób, skłania do kontynuacji majowych spotkań w Świnoujściu.

Kontynuowanie Szkoły w dotychczasowej formule organizacyjnej i merytorycznej wydaje się dalej niemożliwe, jeśli nie chcemy sprawić zawodu uczestnikom kolejnych Szkół Wiosennych. Formuła wykładu trzygodzinnego, uzupełnionego dwoma krótszymi referatami, stawia zbyt duże wymagania dla coraz mniej licznych zespołów gotowych zaprezentować swoje osiągnięcia na tego typu spotkaniach. Zamierzamy zachować wszystko to, co w dotychczasowych Szkołach Wiosennych zyskało aprobatę uczestników oraz wprowadzić nowości, które wpłyną na atrakcyjność Szkoły nie obniżając jej poziomu merytorycznego.

Obok tradycyjnych, zamawianych referatów, planujemy wprowadzenie do programu Szkoły referatów z wolnego naboru. Zwracamy się więc z uprzejmą prośbą o nadsyłanie zgłoszeń referatów wraz z krótkim streszczeniem na adres organizatorów w terminie do końca lutego br. Nie gwarantujemy akceptacji wszystkich zgłoszeń, ale zobowiązujemy się odpowiedzieć na każdą zgłoszoną propozycję.

Tradycyjnie planujemy wydanie materiałów konferencyjnych, w których zostaną zamieszczone teksty wszystkich wystąpień. Wydawnictwo, bardzo popularne nie tylko wśród uczestników Szkoły, ale również rozprowadzane wśród wszystkich zainteresowanych, stanowi bardzo dobry materiał reklamowy dla produkcji niekoniecznie związanych z tematyką Szkoły. Zapraszamy zainteresowanych do umieszczania notek informacyjnych i materiałów reklamowych w materiałach Szkoły. Do współpracy zapraszamy osoby prywatne i firmy informatyczne. Jesteśmy otwarci również na inne ciekawe propozycje podnoszące atrakcyjność imprezy.

Tematyka Wiosennej Szkoły PTI Świnoujście'94 będzie koncentrowała się wokół problematyki: **Języki 4GL**. Pragniemy przedstawić różne rozumienie tego terminu, który na trwałe wszedł do słownika terminów informatycznych, chociaż jest bardzo różnorodnie wykorzystywany. Chcemy przybliżyć nowe projekty, wspomaganie programowania, doświadczenia praktyczne użytkownika języków 4GL, uwarunkowania stosowania, efekty występujące w różnych fazach procesu wytwarzania systemu informatycznego. Mamy nadzieję na zderzenie różnych, często kontrowersyjnych poglądów, co powinno przybliżyć uczestnikom Szkoły rozumienie tego terminu i wyrobienie sobie własnej opinii na temat języków programowania 4GL.

Informacje organizacyjne

Wiosenna Szkoła PTI Świnoujście'94 planowana jest w dniach 23–27 maja 1994 r. Tradycyjnym miejscem spotkania będzie Ośrodek „Dolna Odra" w Świnoujściu. Termin zgłoszeń uczestnictwa mija 30 kwietnia 1994 r. Zgłoszenia te należy przesyłać na specjalnych formularzach, które zostaną rozesłane do zainteresowanych w Komunikacie Nr 2 do 15 marca br.

Adres organizatorów:

Koło PTI w Szczecinie, ul. Mickiewicza 64, 71-101 Szczecin
dr hab. Zdzisław Szyjewski,
tel.: (+91) 87-80-80 oraz (+91) 760-31 w. 319
faks: (+91) 87-88-64, E-mail: zs@plszus11.bitnet

INFOSYSTEM'94

Międzynarodowe Targi Elektroniki, Telekomunikacji i Techniki Komputerowej

17–20 maja 1994 r.

★ sprzęt komputerowy, systemy komputerowe i urządzenia peryferyjne,

★ oprogramowanie systemowe i użytkowe,

★ podzespoły i elementy elektroniczne i teletechniczne,

★ usługi w zakresie oprogramowania oraz wykonawstwa i montażu podzespołów,

★ urządzenia technologiczne dla przemysłu elektronicznego,

★ aparatura kontrolno-pomiarowa i urządzenia automatyki przemysłowej,

★ telekomunikacja.

W 1993 r. w targach wzięło udział 460 wystawców z 13 krajów.

Międzynarodowe Targi Poznańskie
60-734 Poznań
ul. Głogowska 14,
tel.: 69-25-92, 66-43-14,
faks: 66-58-27, 66-07-07,
teleks: 413251

840/93

ASK Ingres Group

Informujemy Państwa, że w dniu 5 marca 1994 r. odbędą się na stoku Nosala w Zakopanem

I Międzynarodowe Zawody Narciarskie

o

Puchar Ingresu.

Po zawodach planujemy bankiet dla zawodników i osób towarzyszących. Opłata startowa 350.000,- zł.
Zgłoszenia prosimy nadsyłać do dnia 28.02.1994 na adres:

RODAN SYSTEM

Ingres

AUTORYZOWANY DYSTRYBUTOR SYSTEMU INGRES
ul.Czerniakowska 73/79, 00-718 WARSZAWA
telefon +48 22 412174, fax : +48 22 414396

Informacji udziela Pani Agnieszka Rowicka

Zgłaszam swój udział w I Międzynarodowych Zawodach Narciarskich o Puchar Ingresu.

_____

_____

imię, nazwisko, adres, telefon

Proszę o zarezerwowanie w dniach 4-6.03.1994 pokoju w hotelu dla:

❒ 1 osoby ❒ 2 osób ❒ proszę nie rezerwować

❒ od 4.03.1994 ❒ od 5.03.1994

Jednocześnie przesyłam kopię wpłaty 350.000,- zł na konto:
Rodan-System Sp. z o.o., ul. Czerniakowska 73/79, 00-718 Warszawa
PKO SA VII O/Warszawa konto nr. 501190-10003978-2511-3-1110

0/1/94